WWW.UPDF.COM

UTRECHT UNIVERSITY

Graduate School of Natural Science

Human Computer Interaction

Keyed Alike: Towards Versatile Domain-Specific Keyword Extraction with BERT

First Supervisor:

Prof. Sergey Sosnovsky

Second Supervisor:

Isaac Alpizar Chacon

Written by:

Lorenzo Pozzi

Academic year 2022/2023

HCI-2869950

UPDF
WWW.UPDF.COM

Acronyms

AKE Automatic Keyword Extraction. 10

BERT Bidirectional Encoder Representations from Transformers. 10

BiLSTM Bidirectional Long-short Term Memory. 12

CRF Conditional Random Field. 16

DSLM Domain-Specific Language Model. 11

DSM Distributional Semantic Model. 18

FFN Feed-Forward Network. 33

FN False Negative. 53

FP False Positive. 53

FPR False Positive Rate. 53

GC Global Corpus. 38

GC+ Global Corpus Plus. 55

GIGO Garbage In Garbage Out. 41

LDA Latent Dirichlet Allocation. 14

LM Language Model. 10

LSTM Long-short Term Memory. 25

MLM Masked Language Model. 34

nDCG normalized Discounted Cumulative Gain. 59

NER Named Entity Recognition. 11

NLP Natural Language Processing. 10

NSP Next Sentence Prediction. 34

PDF Portable Document Format. 18

UPDF
WWW.UPDF.COM

PKE Python Keyphrase Extraction. 57

POS part-of-speech. 16

PR Precision-Recal. 54

PWC Paper With Code. 37

RNN Recurrent Neural Network. 10

ROC Receiver Operating Characteristic. 53

SOTA state-of-the-art. 10

TEI Text Encoding Initiative. 20

TF-IDF Term Frequency–Inverse Document Frequency. 13

TN True Negative. 53

TOC Table of Content. 20

TP True Positive. 53

TPR True Positive Rate. 53

VSM Vector Space Model. 11

UPDF
WWW.UPDF.COM

Terminology

cross-link task A document similarity task in which the goal is to retrieve the most related sections in two documents. 21

cross-validation A resampling method that uses different portions of the data to test and train a model on different iterations.. 39

distributional hypothesis A basic idea of distributional semantics according to which linguistic items with similar distributions have similar meanings. 22

document similarity task An NLP task whose goal is measuring of how similar two documents or groups of texts are to each other. 11

domain class A set of four categories, i.e. core-domain, in-domain, related-domain, or out-of-domain, used to categorize index entries. 21

downstream task A supervised-learning tasks that utilize a pre-trained model. 35

keyphrase A group of words that describes the content of a passage. 10

keyword A word that describes the content of a passage. 10

lexical knowledge Encompasses the information that is known about words and the context in which they appear. 12

metatopic The thematic center of a book. 20

region model A class of Distributional Semantic Models where semantic neighbors are topically similar. 65

self-attention mechanism A machine learning technique that allows the inputs to interact with each other and find out to what they should pay more attention to. 10

semantic domain An area of meaning that contain all the words used to talk about it. 23

sequence labeling task An NLP task which assigns a class or label to each token in a given input sequence. 52

term A word or an expression. 10

WWW.UPDF.COM

vanishing gradient A dispersion of gradient information from the output end of the model to the layers near the input end of the model. 25

word embedding A learned representation for text where words that have the same meaning have a similar representation. 22

word piece A segment of a full word split by WordPiece. 30

word space A vector space populated by word embeddings. 22

UPDF
WWW.UPDF.COM

ABSTRACT

Automatic Keyword Extraction involves the identification of representative terms in a passage of interest. There are various applications, including topic modeling, semantic search, information retrieval, and text summarization where a set of keywords is highly effective. Past research showed the potential of Transformer models for the task. However, such architectures are limited by the need for labeled datasets that require time and effort to be annotated. The present research seeks to overcome this limitation by proposing an alternative annotation approach for Automatic Keyword Extraction corpora, generalizable to diverse domains with reduced costs and production time. Then, a model based on Bidirectional Encoder Representations from Transformers (BERT) will be fine-tuned to extract domain-specific terms from the generated dataset. The experiments aim to corroborate the designed annotation procedure and shed light on BERT's capability in recognizing relevant terms for domain-specific documents. This thesis also proposes an analysis of the word space generated by BERT in order to study the effect of fine-tuning on Automatic Keyword Extraction. The results showed that the proposed solution for dataset annotation was effective and that the implemented BERT-based model outperformed the baselines in all the proposed tasks. Moreover, the final analysis indicates that BERT's word space follows a semantic coherence since the generated embeddings are arranged based on the relatedness to the target domain.

UPDF
WWW.UPDF.COM

Acknowledgements

This research project could not have been possible without my first supervisor Prof. Sergey Sosnovsky and my second Supervisor Isaac Alpizar Chacon who accepted to start this journey with me. Thanks for the numerous meetings and the active contribution without which this thesis wouldn't be completed.

I special thanks go to Sergey Lobachev from the Brookfield Indexing Services and Christine Boylan from the Society of Indexers who gently gave their time to speak with me in interesting interviews about compiling indexes. I also have to thank Melanie Gee, again from the Society of Indexers, and Nicola King for the exchange of emails .

I am also grateful to Deokjin Se, co-author of the paper "Fine-tuning BERT Models for Keyphrase Extraction in Scientific Articles", for the valuable insights into the research field of Automatic Keyword Extraction and the practical advice that he gave me to conduct my experiments.

A debt of gratitude is owed to Denis, assistant professor of Computational Linguistics at Utrecht University, that with his expertise contributed to shaping the connection between BERT and Distributional Semantic Models.

Last but not least, I would like to dedicate this research to my family and friends that stayed close to me and gave me moral support for the whole duration of the project.

UPDF
WWW.UPDF.COM

Contents

UPDF
WWW.UPDF.COM

UPDF
WWW.UPDF.COM

Chapter 1

Introduction

Automatic Keyword Extraction (AKE) concerns the identification of representative and meaningful terms from text data. These terms, also known as keywords or keyphrases[1], describe the content of the passage in which they occur. The ultimate goal of this process is to reduce the complexity of natural language and condense the meaning of a text into fewer terms. In the field of Natural Language Processing (NLP), many applications such as text classification, document clustering, information mining, or web search, often require *document representations* (Liu, Lin, and M. Sun, 2020) to efficiently encode only the essential information. Keywords extracted by AKE algorithms can be used to create such condensed representations, aiding the identification and processing of larger amounts of documents with fewer resources.

The landscape of AKE has undergone deep transformations throughout the years. Three major families of models have been utilized for the identification of relevant terms in text data: (i) *unsupervised learning* systems that approach the problem as a ranking task in which candidate terms go through an initial selection phase and then are sorted based on statistical features derived directly from the document; (ii) *supervised learning* models that make use of labeled data and hand-crafted features to extract relevant terms in a classification fashion; (iii) *deep learning* methods that rely on multiple layers of neural networks to encode the meaning of a text into embedding representations.

A good deal of the most recent research has focused on this last category. More specifically, large-scale pre-trained Language Models (LMs) that rely on the *self-attention mechanism*, also known as Transformers (Vaswani et al., 2017), have been extensively studied in a large body of literature, demonstrating undeniable effectiveness and strong adaptability to a plethora of downstream tasks; two characteristics that make them the current state-of-the-art (SOTA) in NLP. Many researchers have utilized the BERT model (Devlin et al., 2018), short for Bidirectional Encoder Representations from Transformers, to extract domain-relevant terms in unstructured text. The success of this model can be linked to two main factors. Differently from previous Recurrent Neural Network (RNN) based architectures that encode strings of text word-by-word, BERT and other Transformers models rely on self-attention to process the input sequence all at once, making the training phase more efficient. Second, BERT was designed to be purely "bidirectional". Meaning that it can read

[1]To lighten the use of terminology, "keyword" will also refer to the term "keyphrase" from now on.

UPDF
WWW.UPDF.COM

a text all at once, with no specific direction. Thanks to this bi-directionality, the sense of every single word is not represented independently but enriched by its neighboring ones. This novel encoding paradigm allowed the generation of high-quality word representations that, enhanced by the context knowledge, improved the performance in many NLP tasks. Motivated by these achievements and by recent findings on AKE research (Khan et al., 2022), this thesis supports the argumentation that BERT's contextualized embeddings can also improve the identification of keywords in a given passage. Hence, I implemented a BERT-based architecture to extract meaningful terms from text data and use them to construct document representations in the form of Vector Space Models (VSMs) (G. Salton, Wong, and C. S. Yang, 1975): the relevant information is condensed into fixed-length vectors (Y. Zhang, Jin, and Z.-H. Zhou, 2010) where each dimension consist of a term; these representations are then tested on a document similarity task where the goal is identifying the most related sections between two documents.

Despite the dominance of Transformers-based LMs in the field, their hunger for data still represents a restrictive factor: the existence of an ad-hoc corpus is an inevitable prerequisite to match before being able to train any model on a desired task. This aspect is particularly consequential when it comes to Domain-Specific Language Models (DSLMs) (Gu et al., 2021). In contrast to the more diffused general-purpose LMs (Piji Li, 2020), DSLMs are specialized in a particular application domain. Processing domain-specific text requires a deeper understanding of a large number of domain-specific terms that general NLP approaches fail to capture. As an example, if working in the field of Statistics, an algorithm should identify the acronym GLM, which stands for Generalized Liner Model, a term strongly related to the statistical domain.

DSLMs are a powerful tools for document representation and have been extensively used for a variety of domain and tasks, including Question Answering tasks in Biomedical research (Gu et al., 2021); Named Entity Recognition (NER) in the architecture, engineering, and construction domain (Zheng et al., 2022); financial Sentiment Analysis (Araci, 2019); and identifications of breast cancer phenotypes from electronic health records (S. Zhou et al., 2022). The diversification in this branch of research is, however, hindered by the high costs of dataset annotation which require prepared and well-paid domain experts.

The present research seeks to overcome this limitation and harness the power of SOTA Transformers models for the extraction of keywords with a threefold aim:

1. *propose an alternative to manually annotated domain-specific corpora.* I implemented a versatile pipeline for the construction of AKE corpora that requires minimal human intervention and costs of production. Such a pipeline was designed to be applied in a diversified number of domains, independently from the focus of the field of interest.

2. *uncover the potentialities of pre-trained LM on a closed-domain scenario* and probe their efficiency in extracting relevant terms for the generation of VSMs. A BERT-based model was used to retrieve keywords from domain-specific documents with the goal of identifying the most related sections. An in-depth analysis was performed to discuss the major flaws and benefits of these models for the given task.

WWW.UPDF.COM

3. *contribute to the current state of knowledge about how BERT works* and in particular study the effect that fine-tuning on a specific domain has on the model behavior. The hypothesis is that not only BERT learns to recognize patterns from the pool of training samples, but it also relies on acquired lexical knowledge, generating word embeddings whose location in the vector space reflects the closeness with the target domain.

With a curious sentiment, this study sets out to challenge large-scale pre-trained LMs and investigate their usefulness in an application where, to our knowledge, they were scarcely tested. Hence, I explored their ability to capture lexical patterns and identify key terms in a long-length document while being aware of its domain-specificity. Given the interdisciplinary of this research, Chapter 3 provides some background knowledge on the topic and more thoroughly define a few central concepts. The subsequent chapter gives an overview of the current landscape in the research of AKE techniques: §2 is split into three main sections, each dedicated to uncovering the principal characteristics of unsupervised, supervised, and deep learning method for key term identification, respectively. Chapter 5 is concerned with the construction of the dataset used for this study. In an attempt to contrast the paucity of domain-specific corpora, an end-to-end pipeline for the annotation of domain-specific texts is here described. Next up is the formal definition of our composite model (§6): I opted for a BERT encoder combined with a Bidirectional Long-Short Term Memory (BiLSTM) module and a post-filtering block that helps maintain consistency across the predictions. Chapter 7 analyses the results of the framework and concludes with an empirical evaluation that seeks to better understand the black box nature of BERT through a spatial exploration of the generated embeddings before and after fine-tuning. Lastly, §8 and §9 discuss the significance of the findings for the research on pre-trained LMs, illustrate some limitations, and close with future efforts to improve on the present work.

UPDF
WWW.UPDF.COM

Chapter 2

Overview on Automatic Keyword Extraction Research

AKE deals with the automatic recognition of keywords from unstructured text. Traditionally, AKE systems have been using a two-step approach. First, “candidates” keywords are identified based on their representativeness of the document, and subsequently, whether the approach is supervised or unsupervised, elements in the list are respectively classified or ranked using various strategies and features. These features can be derived from external resources (e.g. Knowledge Bases) or generated from the document itself, expressed by statistical, structural, or linguistic properties.

In this chapter, I follow the structure proposed by (Papagiannopoulou and Tsoumakas, 2020) and (Alami Merrouni, Frikh, and Ouhbi, 2020) that distinguished the most prominent extractive techniques into three categories: unsupervised, supervised, and deep models.

2.1 Unsupervised learning

As previously stated, unsupervised methods regard AKE as a ranking task in which candidate words that pass the initial filtering phase are sorted based on features borrowed directly from the specific document or the corpus of documents. I present here an overview of the most relevant approaches and techniques for unsupervised term extraction tasks.

Statistics-based Methods One of the most popular unsupervised metrics used in a variety of information retrieval tasks is TF-IDF, short for Term Frequency–Inverse Document Frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. BM25 (Robertson, Zaragoza, et al., 2009) improves on this using a ranking function that takes into account document length and term relevance saturation to estimate the relevance of a word to a given document. KP-miner (El-Beltagy and Rafea, 2009) implements it to compute terms valence in a collection of documents. Improving on this work, KeyCluster (Liu, Peng Li, et al., 2009) introduces co-occurrence metrics: the algorithm groups candidate phrases in clusters and identify an exemplar term representative of each cluster based on semantic relatedness. The principle behind this approach is that these exemplar terms should give a good approximation of the content of the document. YAKE (Campos et al., 2020) represents another important step forward in

UPDF
WWW.UPDF.COM

AKE research. The algorithm rests on statistical and co-occurrence metrics but also new metrics that capture context information of the candidates.

Although statistic-based methods were initially the most popular approach, they are unable to deal with big datasets as they suffer from information overload: a large number of text data may cause the statistical information to be noise, and the real meaning of the words in the document to be distorted (Papagiannopoulou and Tsoumakas, 2020). Moreover, they do not take into account semantic similarities: if two documents have similar meaning but they are of different words, similarity cannot be computed effectively.

Graph-based Methods This family of unsupervised methods aims to represent the target document as a graph where each key phrase is assigned with a node. Here, the keyphrase extraction task consists of sorting the nodes by the number and weight of their edges that are directly connected to the importance of the phrase. This technique was proposed by Google with PageRank (Page and Brin, 1998) to order online search results. TextRank (Mihalcea and Tarau, 2004) was the first algorithm that applied it for keyphrases extraction but many other studies were built upon it, leading to successful methods such as the ones proposed by (Wan and Xiao, 2008) and (Florescu and Caragea, 2017).

The main disadvantage of the above-cited approaches is that they consider the documents in a collection to be independent from each other, meaning that only the information in the target document is considered for the extraction of key-phrases. However, documents can be related to each other if belonging to the same domain. ExpandRank (Wan and Xiao, 2008) works around this fallacy and assigns a TF-IDF score to each document in the collection and groups the k most similar ones to the target in the same cluster. After the generation of this knowledge context around the target document, a graph is created with the candidates from each document, and edges are added between two nodes that co-occur in the same window of words in the document set.

Some approaches attempt to extract key phrases related to topics discussed in the document through clustering techniques or Latent Dirichlet Allocation (LDA). TopicRank (Bougouin, Boudin, and Daille, 2013) relies on topical representations of the document that are first weighed and then used to generate key-phrases. The advantage of this approach comes from the intuition that ranking topics, defined as a cluster of similar expressions, are a more straightforward way to identify the set of key-phrases that cover the main topics. Motivated by the fact that a document can be represented by different topics, Topical PageRank (Liu, Huang, et al., 2010) uses LDA to compute the corresponding topic distribution. Subsequently, PageRanks is applied to calculate the importance scores of words under different topics. While statistics-based approaches may suffer from the overwhelming amount of processed data, the issue with topic-based graph methods is that they are exposed to *information loss*. If two words never co-occur in the same context there will never be a connection in the graphs representing the document, even though the two terms are semantically correlated.

Other studies (Shi et al., 2017) account for this weakness by clustering candidate terms using semantic information from external knowledge graphs such as DBpedia. Although this showed notable improvement compared to past research, the sheer introduction of semantic relation information is not sufficient to create background knowledge. (S. Wang et al., 2015) propose an improved PageRank model that implements pre-trained word embeddings to weight connections between two edges. Hence, the algorithm computes the strength of relation taking into account the product of semantic relatedness.

Keywords Extraction with Embeddings Similarly, many keyword extraction use sentence embeddings, i.e. vector representation that capture the semantic information of entire phrases. EmbedRank (Bennani-Smires et al., 2018) implements a simple approach and compares the semantic vector of the target document and each candidate entry. Therefore the ranking phase is carried out by computing the similarity between these two embeddings.

Language Model-based Methods An different approach was proposed by (Tomokiyo and Hurst, 2003) where an N-gram language models are used for key-phrase extraction. The idea is to train multiple language models both on a foreground corpus (target document) and a background corpus (document set) for using pointwise KL-divergence to score phraseness and informativenes of the extracted phrases.

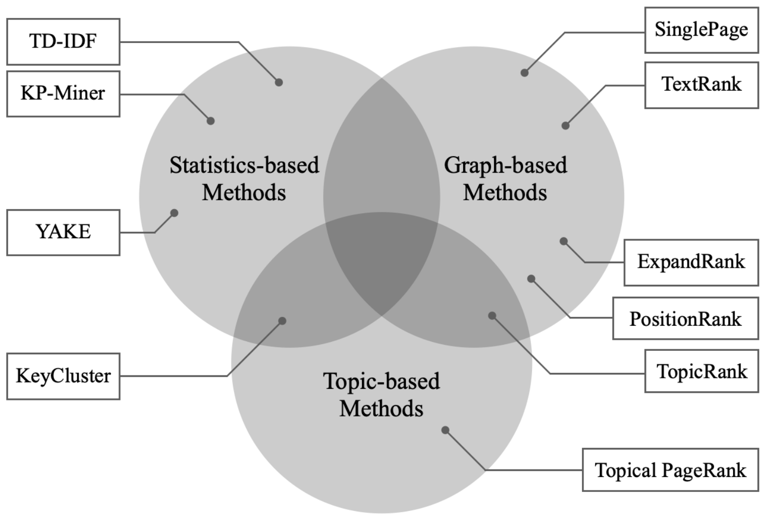


FIGURE 2.1: Models often implement more than one strategy to identify key-phrase in the document. The graph represents the most common unsupervised models according to the technology applied.

2.2 Supervised learning

In supervised approaches, feature selection is a fundamental phase in which relevant predictors are manually selected to train the model. The features commonly implemented in supervised methods can be divided into two groups: within-document features and out-of-document feature.

Withing-document features are derived from characteristics of the text being inspected. These features can be further specified into four categories:

- *Statistical features* are computed based on statistical information gathered from the training documents. TF-IDF and BM25 are well-known scores utilized in many studies.

UPDF
WWW.UPDF.COM

- *Structural features* encode how different candidate key-phrases are located in different parts of a document. For example, terms that appear in titles or at the beginning of a paragraph are generally more representative of the document.

- *Linguistic features* encode the syntactic patterns of a candidate key-phrase. Numerous empirical studies proved the usefulness of part-of-speech (POS) tags and suffix sequences to identify key-phrases.

- *Context features* capture the meaning of a term based on surrounding words, under the assumption that the context of a word shows its meaning.

Conversely, *out-of-the-document features* are extracted from external corpora, such as knowledge bases (e.g. Wikipedia). The information derived from a corpus can be of various natures: some studies rely on the presence of the key phrase in the ontology others on pre-trained word embeddings.

Once useful feature are identified, the task itself can be reformulated in different ways. Traditionally supervised methods treat this task as a binary classification problem where the goal is to train a model that maps candidates into two classes: "key-phrase" and "not-key-phrase". Models like KEA and its posterior extensions make use of a Naive Bayes learning algorithm and showed noticeable performance (Witten et al., 1999), (Caragea et al., 2014). In contrast with this current of thought, (Hasan and V. Ng, 2014) points out that these methods treat candidate terms identically, overlooking the natural difference between them. Indeed, not all key phrases represent a topic in the same way and therefore more representative terms must be preferred.

Motivated by this observation, rank approaches such as Ranking SVM (Jiang, Hu, and H. Li, 2009) learn a function that sorts the candidates based on their relevance score. Researchers also reframed keywords extraction as an optimization problem of the ranking function, where convergence is found with the optimal set of weights (Yu and V. Ng, 2018).

A more recent line of research relies on a sequence labeling task to identify keywords in text. (Gollapalli, X.-L. Li, and P. Yang, 2017) trained a tagger with Conditional Random Fields (CRFs) which achieve performance comparable to older SOTA models but using within-document features alone.

2.3 Deep Learning Methods

Although deep learning approaches belong to the family of supervised learning, they neither rely on the traditional two-step workflow nor manually-defined features. Conversely, these two phases are simultaneously modeled into a deep sequence of neural layers that compress the semantic information of the input sequence into a dense numerical vector. Among the earliest researches, (Q. Zhang et al., 2016) proposed a RNN model to extract keywords reformulating the problem as a sequence labeling task. All the previous models were strictly following the paradigm of AKE since they output a list of keywords coming from the target document. In (Meng et al., 2017) authors propose a generative model that leveraging the encoder-decoder framework which can potentially generate key phrases that do not appear in the source text. This is possible because in encoder-decoder the length of output sequence is independent from that of the input, given that the former is generated

WWW.UPDF.COM

from dense representations of the original text. Subsequent works further improved on the generative approach by integrating a reinforcement learning reward function (Chan et al., 2020) that encourages the model to generate both sufficient and accurate key phrases.

An investigation of the literature revealed few studies that focus on Transformer-based models for AKE task. Unlike most other proposed state-of-the-art neural keyword extractors, (Sahrawat et al., 2020) did not employ recurrent neural networks but instead used the contextualized word embedding generated by pre-trained transformer models and feed them into a BiLSTM network paired with an additional Conditional random fields layer (BiLSTM-CRF). This study demonstrates how contextual generated by BERT embeddings significantly improved on other encoder models and the applicability of such a family of models for key terms extraction tasks. Another transformer-based approach is TNT-KID (Martinc, Škrlj, and Pollak, 2021). Here the authors explored transfer learning techniques and studied how these affect the performance of the keyword extractor.

2.4 Discussion on Available Approaches

The above-presented models have strengths and weaknesses and might be more suitable depending on the final application and resources at one's disposal. Given the increasing amount of data that is to be processed, unsupervised approaches come with the benefit of being completely independent from datasets. In niche domains, in fact, there might be a paucity of data available to train a model. For instance, in the case of minority languages, transfer learning techniques are often the only solution to work around the problem. Graph-based methods are considered the state-of-the-art for unsupervised models (C. Sun et al., 2020) as they take into account global information computed from the entire graph, rather than relying only on local information. This procedure captures the essential notion behind a keyword. Indeed, the importance of a candidate term in a document comes from the degree of relatedness that it has with other candidates. Graph-based methods model this idea with weighted connections between two nodes. Despite the benefits, this configuration has one major flaw: given that a set of key-phrases should ideally cover the main topics in the document, such models do not guarantee this assumption as they are inherently biased toward terms that describe more prevailing topics. Other than a weak topic coverage, unsupervised methods also fail to adapt to the specifics of the syntax, semantics, content, genre, and keyword assignment regime of a specific text, thereby showing lower performance compared to supervised methods.

Pre-trained LMs bested previous neural network architectures, such as RNN or LSTM, in all standard NLP tasks while their advantage over past approaches has been extensively proven in numerous empirical studies. Among this family, BERT has shown a superior ability in capturing syntactic and semantic relationships within the input sequence and generating high-quality word embeddings applicable to a great number of downstream tasks.

Given their greater ability in understanding natural language compared to both unsupervised methods and past supervised models, I decided to experiment with pre-trained language models, studying their effectiveness in an AKE task.

UPDF
WWW.UPDF.COM

Chapter 3

Background Concepts

Before continuing with the description of BERT, the reader is introduced here with some auxiliary concepts that s/he should be aware of to better follow the present research.

The discourse starts with a broad depiction of what a textbook index is. Indexes have a pivotal role in our annotation pipeline. Inspired by (Alpizar-Chacon and Sosnovsky, 2020), I saw in the domain-specificity of textbooks and the organized structure of indexes an opportunity to extract knowledge-specific information apt for the automatic construction of AKE corpora. For this reason, section §3.1 describes the main elements of indexes, which will recurrently appear throughout the coming pages.

As expressed in the second contribution, the developed pipeline is designed to be applicable to many domains. To do so, it is grounded in the increasing number of academic textbooks available online in the Portable Document Format (PDF). Although these digital resources are virtually unlimited, PDFs contains a variety of data types (e.g. text, tables, images, diagram) that are not directly accessible because non editable. Hence, an intermediate tool was needed to automatically extract the information contained in these documents. A specialized system developed for textbook parsing was used for the processing of PDFs. Section 3.2 outlines the main features of the parser. This information was then used to label a dataset for AKE tasks, working around the manual annotation process, and reducing the time and costs of production.

Once the dataset is constructed and the model trained on the task, a similarity metric is calculated between term vector representations, i.e. VSMs, of two different documents. VSMs already received a brief introduction in Chapter 1. Section 3.3 further elaborate on them giving a more formal definition and contextualizing their use to one of the evaluation tasks.

Lastly, I discuss the connection between BERT and the family of models known as Distributional Semantic Models (DSMs). Emphasizing on the effect that fine-tuning BERT for AKE has on its embeddings.

3.1 Textbook Indexes

Book indexes were first introduced as a form of navigational tool to help readers orient in text documents. N.C. Mulvany, author of *Indexing Books* (Mulvany, 2013), gives an exhaustive definition: *an index is a structured sequence, resulting from*

UPDF
WWW.UPDF.COM

a thorough and complete analysis of text, of synthesized access points to all the information contained in the text. The structured arrangement of the index enables users to locate information efficiently.

From this description, it is clear that the value of a good index does not come solely from the cited terms and respective locators but also in the knowledge structure in which they are organized. This structure presents relevant terms in an arrangement that reflects their hierarchical relationship and it does so in a way that is as transparent as possible to the reader.

Conventionally, once a book is in its final form, third-party professionals are entrusted to compile the index by distinguishing between relevant and peripheral information, indicating relationships between key concepts, separating them into main entries and subcategories, and creating cross-references. FIGURE 3.1 shows the final result of this process, highlighting the main elements that compose a good-quality index.

Main headings or *subject headings* are the primary access point in the index. Indeed, readers conduct their search for information at the main heading level. For this reason, indexers must adopt terms that cover the most relevant themes in the document and at the same time see through the readers' eyes and opt for entries that they will be likely to look up.

The lines of indented text that follow the main heading are called *subheadings*. Subheadings, also called by some authors subentries, refer to lower categories that belong to the topic described by the respective main subjects. Depending on the focus of the analysis, indexers may break major topics in different ways. For instance, in a book about classical engines the main heading "vehicles" would be with subentries "Honda", "Kawasaki", and "Ford" if the brand is the focus, or "electric" and "diesel" if the analysis is on the type of engine. When a hierarchical relationship emerges in an index through the use of subheadings, the indexer carefully evaluate the nature of this relation according to the specific purpose of the entries.

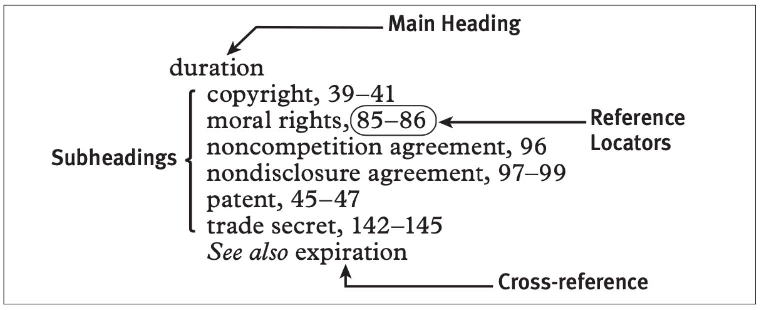


FIGURE 3.1: Snippet of a back-of-the-book index from *Indexing Books* of N.C. Mulvany. Main headings represent a top-level term that establishes the shared meaning in the same group of sub-entries. Reference locator and cross-references are directional tools that orient readers between the book sections and index entries respectively.

Reference locators indicate the segment of the document where the referred entry can be found. However, the location does not always represent the page of interest: locators can also guide the reader to the section number.

Cross-references are used as navigation tools to relate heading in the index that shares close meaning. They can be found in two different forms. Indexers may decide for entries that do not appear explicitly in the source document, the *See*

UPDF
WWW.UPDF.COM

cross-reference directs readers from a term not used in the index to the term that is used, for instance “vehicles. *See* automobile”. The *See also* cross-reference directs readers to closely related index entries, “spam filtering, 3, 83. *See also* adversarial machine learning” is an example.

With that said, I believe two factors make it possible to use indexes for labeling domain-specific corpora. While indexing a book, the annotators always take into account the so-called metatopic, the central matter discussed. Consequently, the great majority of the index entries are related to the target domain. Second, professional indexers are experts in the field for which they are hired. Identifying indexable topics and deciding how to present an index require them to have a high level of reading comprehension, classification ability, and conceptualization skills, particularly regarding thematic cross-linking. All competencies that take years of training to develop. This expertise guarantees the quality of the produces indexes and of the index terms included in them.

Hence, the annotation of the documents proceeds as follows. Each academic book included in the dataset is labeled identifying all the occurrences of the corresponding index terms. More details on the dataset construction and annotation in §5.

3.2 Digital Book Parsing

PDF Parsers are tools that allow the extraction of text data from PDF files. A PDF document can contain various types of information, including text, fonts, vector graphics, images, etc. Despite their versatility to many softwares, PDF documents contains unstructured information that has to be extracted and pre-processed before being used in other applications. Parsers replace the traditional manual data extraction process by identifying the non editable components of a PDF file, transforming it into a structured format.

In (Alpizar-Chacon and Sosnovsky, 2020), the authors explored textbooks as a case study and developed a sophisticated system of heuristic rules to capture diverse elements in didactical books’ formatting and structure. The final representation of each book was adapted to Text Encoding Initiative (TEI) standards. Following P5 Guidelines [1], they mapped all the sections in a textbook into TEI elements and further grouped this information into three main categories: Structure, Content, and Domain Knowledge.

Structure. A single textbook is divided in a hierarchical way where broader and primitive thematic is described in the first sections while more specific ones are found later in the book. The distribution of sections follows the division decided by the author which is expressed in the Table of Content (TOC). A TOC is a list of the main subjects and subheadings of the document; hence, each section is uniquely identified by its corresponding heading.

Content. If the TOC gives a general structure that distinguishes between the various sections of a textbook, it is also relevant to identify the structural and formatting information contained in each section: pages, fragments, paragraphs, lines, and words are assigned to their corresponding sections identified by a unique heading title.

Domain Knowledge. Lastly, all main and subheadings in the index of the book are

[1] https://tei-c.org/guidelines/p5/

utilized to create a specific VSM of the document. Moreover, each index term is enriched with additional domain information: entries are labeled to one of four domain classes, i.e. *core-domain*, *in-domain*, *related-domain*, or *out-of-domain*, depending on the relatedness that they have with the topic of the book (Alpizar-Chacon and Sosnovsky, 2022); if identified, links to DBpedia pages are also associated to each entry, the links are used to assign one of the above-cited classes to the index term; all the instances of a unique concept in the index are assigned to their corresponding page, together with the sentence where they were extracted.

All the textbooks included in the corpus were parsed with this method and the obtained representations used as the backbone for the construction of the dataset. The main advantage of the above-described system is that it generalizes to any domain, with the only requirement that the textbooks must be accompanied by an index.

3.3 Vector Space Models

VSM is a vectorial model for representing text documents as numerical vectors of identifiers. These document representations have been used in numerous applications such as information retrieval (Eminagaoglu, 2022), relevancy rankings (Lee, Chuang, and Seamons, 1997), and text similarity (Shahmirzadi, Lugowski, and Younge, 2019).

According to (G. Salton, Wong, and C. S. Yang, 1975) and their study on automatic indexing, a document of interest $\boldsymbol{d}_i$ can be rendered in a multi-dimensional vectorial space where each index term add a dimension inside the representation. Formally it can be expressed as follow:

$$\boldsymbol{d}_i = \{t_{i1}, t_{i2}, ..., t_{in}\}$$

where $\boldsymbol{t}_{ij}$ is a weight representing the relevance of the term. Given that each of these terms is translated into a numerical vector, collectively, they contribute to creating a unique representation of the document. This representation allows for a comparison between documents through the computation of a score $s(\boldsymbol{d}_i, \boldsymbol{d}_j)$ that reflects the degree of their similarity.

FIGURE 3.2 illustrates how it would be possible to transpose documents in a vectorial space using VSMs representations. In this space, each document is allocated in a specific region depending on its feature vector; consequently, similar documents are placed closely, forming clustered groups of similar items. Such distribution is not discrete but continuous. Documents may contain elements that commune with others and therefore different clusters may overlap in some contact points. Moreover, for bigger groups, there might be smaller nested structures containing sub-genres of the major group. This organization ensures a high precision search output: a given relevant document is retrievable without retrieving a number of nonrelevant items which will stand far from its location. Conversely, in case multiple documents are returned after a query, such items will belong to the same general area and therefore share an arbitrary degree of similarity with the top recommendation.

The effectiveness of VSMs was put under examination in (Alpizar-Chacon and Sosnovsky, 2020) with the so-called cross-link task, first introduced by (Guerra, Sosnovsky, and Brusilovsky, 2013). As already mentioned in §3.2, the authors generated vector representations using terms included in book indexes. They compared

UPDF
WWW.UPDF.COM

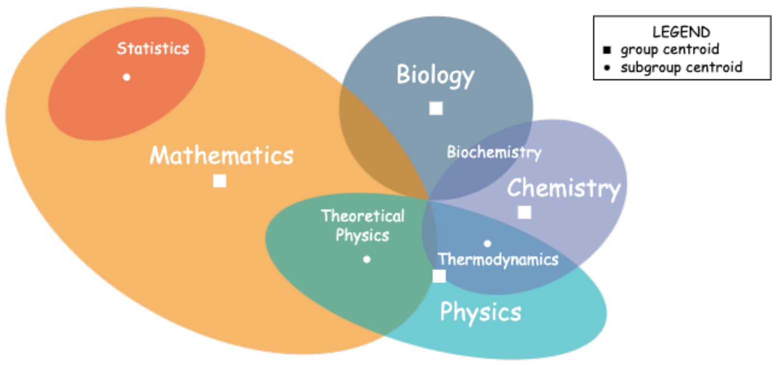


Figure 3.2: An example of clustered document space where the various document are grouped based on their similarity. Clusters may share some portion of the continuous space or be internally organized in smaller agglomerates.

classical TF-IDF and LDA approaches to the constructed VSMs and computed similarities between sections. The objective was that of creating links between books sections that present similar topics. The former approach showed to consistently outperform all the baselines. Echoing the experiment proposed by the above-cited studies, this thesis seeks to improve on past research generating VSMs with the terms extracted from a BERT-based model, and so demonstrate the efficiency of BERT in the cross-link task.

3.4 Vector Semantics and Distributional Semantic Models

Distributional semantics (Lenci, 2018) is a branch of computational linguistics that focuses on the analysis of the lexicon. Its primary aim is to encode the meaning of words in multi-dimensional vectors also known as word embeddingss. Word embeddings are a type of representation where individual words are expressed as real-valued vectors that allow related words to be close in the vector space, called word space. These representations are created by empirical approaches, namely DSMs, based on a statistical analysis of the context in which the word occurs. In other words, the distributional vector captures words' usage i.e. the contexts in which are more likely to appear. The theoretical validity of this approach is supported by the distributional hypothesis, according to which similar words tend to occur in similar linguistic contexts.

The closeness or similarity between word embeddings in such word spaces is base on linguistic features learned during training. Example of a linguistic feature is *paradigmaticity*: two lexemes hold a paradigmatic relationship if they can be interchanged in the same sentence without distorting the general idea behind the phrase. If in the sentence "The tuna swims inside the ocean", the word "tuna" is replaced with "shark", the idea is unaffected as they belong to the same animal family. Conversely, in "The horse swims inside the ocean", something is immediately

WWW.UPDF.COM

perceived as out of place. In the same way humans assume ”tuna” and ”shark” to be more closely related than ”tuna” and ”horse” based on their natural habitats, a DSM would produce word embeddings that respect the similarity between these words depending on the context where they are mostly found.

BERT seems to possess this property. Although it was not solely implemented for lexical representation, (Lenci et al., 2022) include BERT-like algorithms into the family of DSMs. Like standard DSMs, in fact, BERT generates embeddings as a function of the distributional properties of words in the text.

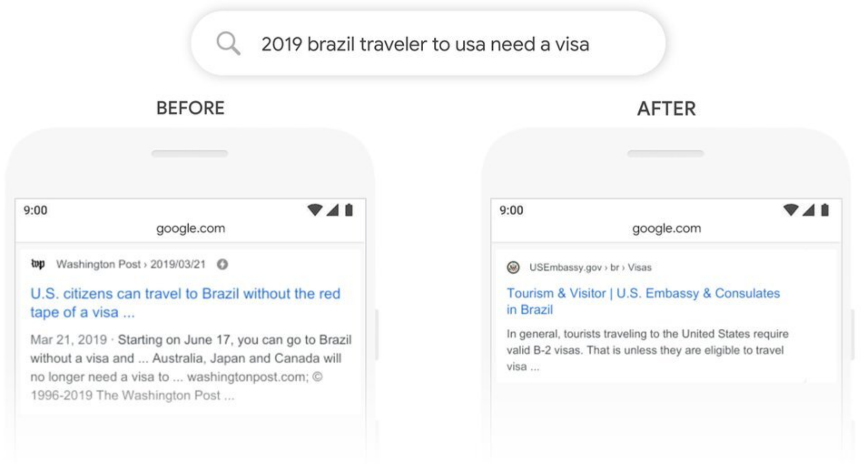
The association of BERT with DSMs is still under debate but is getting stronger among the scientific community. In accordance with this hypothesis, the present study treats BERT as a DSM and analyzes its effectiveness to capture a specific lexical property, i.e. the semantic domain. A semantic domain is a particular area of vocabulary that shares a set of meanings. In the case study examined by this thesis, each domain is identifiable by an academic field. Statistics is an example of a semantic domain. Hence, following the logic of the tune/horse example, the argument is that after being fine-tuned on a domain-specific dataset, BERT captures more defined semantic similarities, generating a word space where domain-related terms lie in closer regions with respect to out-of-domain ones. In linguistics and lexical semantics, this capability is known as lexical knowledge.

UPDF
WWW.UPDF.COM

Chapter 4

BERT

BERT (Devlin et al., 2018) is a deep contextual language representation model belonging to the family of architectures known as Transformers. Language has historically been difficult for machines to "understand" since they lack basic language context. BERT is designed to help machines understand the meaning of ambiguous language in text by generating word embeddings that capture contextual information in the text. FIGURE 4.1 shows the improvement in language understanding by the Google Search engine after integrating the model.


2019 brazil traveler to usa need a visa

BEFORE

9:00
google.com

Washington Post › 2019/03/21

U.S. citizens can travel to Brazil without the red tape of a visa ...

Mar 21, 2019 · Starting on June 17, you can go to Brazil without a visa and ... Australia, Japan and Canada will no longer need a visa to ... washingtonpost.com; © 1996-2019 The Washington Post ...

AFTER

9:00
google.com

USEmbassy.gov › br › Visas

Tourism & Visitor | U.S. Embassy & Consulates in Brazil

In general, tourists traveling to the United States require valid B-2 visas. That is unless they are eligible to travel visa ...

FIGURE 4.1: An example proposed by (Nayak, 2019) where BERT (on the right) has helped Google Search to understand the nuances of language compared to past approaches (on the left). In the query "2019 brazil traveler to usa need a visa.", the word "to" and its relationship to the other words are particularly important to understanding the meaning of the sentence. It's about a Brazilian traveling to the U.S. and not the other way around. The algorithm on the left does not understand the importance of this connection, returning results about U.S. citizens traveling to Brazil. Conversely, BERT is able to grasp the relevance of "to" despite its frequent use in language.

BERT was engineered to pre-train deep bidirectional representations of words from unlabeled text by jointly conditioning on both the left and right contexts. In a transfer learning fashion, the pre-trained model can be further fine-tuned for a wide range of NLP tasks. Leveraging the linguistic features learned during the first stage, fine-tuning results in a much lighter procedure.

UPDF
WWW.UPDF.COM

This chapter explains the importance of BERT in the field and presents its architecture. In particular, Section 4.2 gives a brief overview of previous approaches used in NLP. Continuing from this, Section 4.2 presents Transformers and their innovation with respect to such methods. Then, Section 4.3 describes the BERT model itself, its architecture, parameters, and mechanisms. Finally, Section 4.4 and 4.5 give details about the pre-training and fine-tuning procedures, respectively.

4.1 Before BERT: RNNs, LSTMs, and Bi-LSTMs

Language is an inherently temporal phenomenon. When we comprehend and produce spoken language, we process continuous input streams of indefinite length. An RNN is any network that contains a cycle within its network connections that captures such temporal nature. As FIGURE 4.2 shows, the recurrent link that processes the input $\boldsymbol{x}_t$ in the hidden layer $\boldsymbol{h}_t$ also takes into account the value of the hidden layer from the preceding step $\boldsymbol{h}_{t-1}$. In this way, it encodes context information from earlier processing. This information is then used to enrich the computation at later points in time.

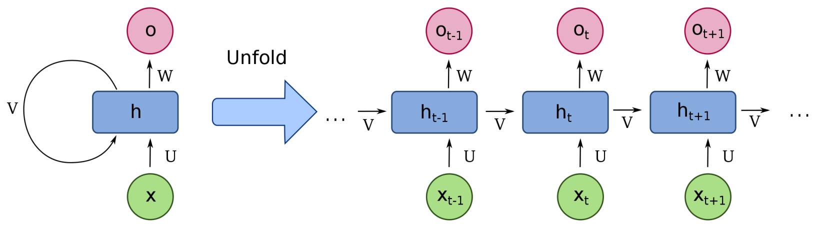


FIGURE 4.2: On the left, is a compressed diagram of an RNN block. On the right, the same representation unfolded in different steps in time (Wikimedia, 2017b)

.

Long Short-Term Memory networks, usually shortened to LSTM, are a special kind of RNN. They were introduced to avoid the vanishing gradient problem that causes the gradients used to update the parameters of a neural network to diminish with the depth of the latter, effectively preventing the weights from changing their values. For RNNs, this means that the longer the sentence the more information is lost in the final steps of the encoding phase. LSTMs try to mitigate this problem with a gated mechanism that regulates the information flow into and out of the so-called memory cell. Specifically, this is done with three gates. The Forget Gate controls what information is to be remuve and decides how much of the past the network should remember. The Input Gate controls what new information is added to the cell state from the current input. Output Gate conditionally decides what to output from the memory. An LSTM block will look like the image below (FIGURE 4.3a).

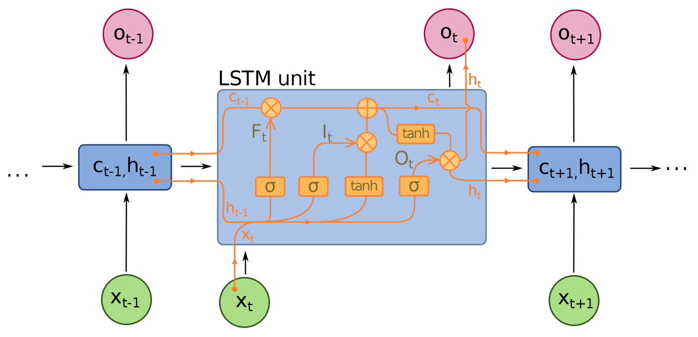
At a time t, an LSTM cell consists of the following modules: ($\boldsymbol{i}_t$) input gate, ($\boldsymbol{o}_t$) output gate, ($\boldsymbol{f}_t$) forget gate, ($\boldsymbol{c}_t$) memory cell, ($\widehat{\boldsymbol{c}}_t$) candidate memory cell. The

UPDF
WWW.UPDF.COM

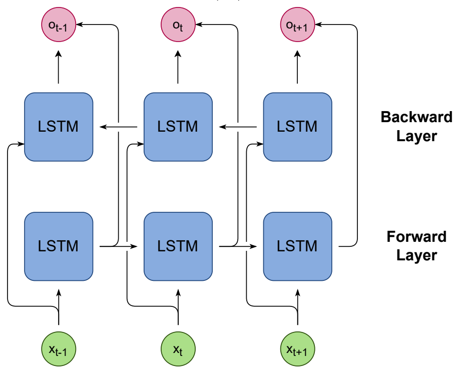
updates are then:

$$
\begin{aligned}
\boldsymbol{i}_t &= \sigma(\boldsymbol{W}_i\boldsymbol{x}_t + \boldsymbol{U}_i\boldsymbol{h}_{t-1} + \boldsymbol{b}_i)\\
\boldsymbol{f}_t &= \sigma(\boldsymbol{W}_f\boldsymbol{x}_t + \boldsymbol{U}_f\boldsymbol{h}_{t-1} + \boldsymbol{b}_f)\\
\boldsymbol{o}_t &= \sigma(\boldsymbol{cW}_o\boldsymbol{x}_t + \boldsymbol{U}_c\boldsymbol{h}_{t-1} + \boldsymbol{b}_c)\\
\widehat{\boldsymbol{c}}_t &= tanh(\boldsymbol{W}_c\boldsymbol{x}_t + \boldsymbol{U}_c\boldsymbol{ch}_{t-1} + \boldsymbol{b}_c)\\
\boldsymbol{c}_t &= f_t \otimes \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \otimes \widehat{\boldsymbol{c}}_t\\
\boldsymbol{h}_t &= \boldsymbol{o}_t \otimes tanh(\boldsymbol{c}_t)
\end{aligned}
\tag{4.1}
$$

Here, σ is the sigmoid function, $\otimes$ is the Hadamard product, $\boldsymbol{x}_t$ is the input vector at time t, and $\boldsymbol{h}_t$ is the hidden state that stores sequential information up to time t. $\boldsymbol{W}$, $\boldsymbol{U}$, and $\boldsymbol{b}$ are model parameters to be estimated during training.



(A)



(B)

FIGURE 4.3: In (A), an unfolded representation of the LSTM architecture taken from (Wikimedia, 2017a). In (B), an illustration of an unfolded BiLSTM (B)

In a BiLSTMs (FIGURE 4.3), the sequence information is processed in two directions: backward and forward. The outputs of the two networks are then combined into a single representation that captures both the left and right contexts at each point in time. The importance of such bidirectional encoding can be easily understood with an example. Give the three incomplete sentences:

- I am ___.
- I am ___ tired.
- I am ___ tired, could we rest now?

UPDF
WWW.UPDF.COM

The blanks can be filled with different words depending on the amount of information that is at disposal, e.g. “happy”, “not”, and “very”. The more context is available the more precise the prediction will be.

4.2 Transformer Models

BERT belongs to the class of deep architectures called Transformers. First described in the well-known *”Attention is all you need”* (Vaswani et al., 2017) from Google, Transformer models represent one of the most powerful classes of models invented to date. Such architectures were initially designed for machine translation to solve the issues of their precursors, i.e. RNNs, LSTM, and BiLSTM.

Traditionally, the architectures applied in this task have two main sections: an encoder and a decoder.

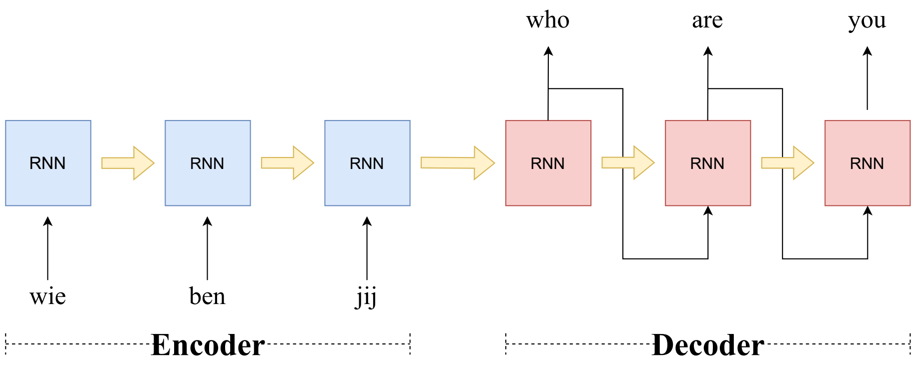


FIGURE 4.4: An illustration of the RNN Encoder-Decoder Model

FIGURE 4.4 shows an encoder-decoder architecture based on RNNs used for translating a sentence from Dutch to English. In this case, the input text is processed sequentially one word at a time: each step makes a computation taking into account the previous representations. When the final step is reached, the encoder generates a vector representation that captures the meaning of the whole input sequence and passes it to the decoder. This, following a similar sequential procedure, generates an output sequence according to the target task. Given the nature of such a processing approach, there is no possibility for parallel computation. This means that the more words are in the input sequence the more time will be required to process the sentence. Another issue with long sequences is that the information from the initial steps tends to get lost due to the vanishing gradient effect.

Transformers models (FIGURE 4.5) were developed to address these problems. First of all, they are not based on RNNs but make use of the self-attention mechanism which avoids sequential computation and allows the processing of text input in parallel. Only a single step is needed for the whole sequence. Moreover, since Transformers do not involve any neural recurrent structure, they are not subjected to vanishing gradients. Lastly, they encode word meaning in a truly bidirectional fashion improving on the approach used by BiLSTM. As pointed out by (Devlin et al., 2018), such architectures are not deeply bidirectional but rather combine the output of the forward LSTM and the backward LSTM in a single representation. In contrast, Transformers do this simultaneously thanks to the attention scores.

UPDF
WWW.UPDF.COM

In short, thanks to the novel self-attention mechanism and a non-sequential approach, Transformers parallelize the manipulation of the input sequence allowing for more intensive training phases on big amounts of data without prohibitive training time.

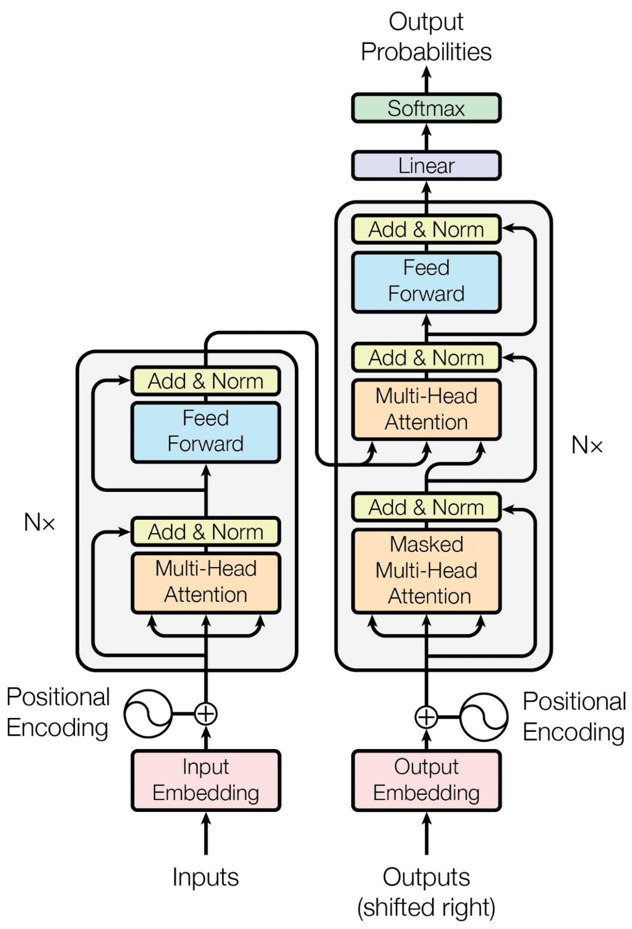


FIGURE 4.5: The encoder-decoder structure of the Transformer architecture taken from "Attention Is All You Need" (Vaswani et al., 2017)

4.3 BERT Architecture

BERT was created from 12 stacked encoder modules of the original Transformer architecture. As shown in FIGURE 4.6, BERT's architecture is composed of different subsystems. An Embedding Layer that prepares the input sequence to be processed. The all-important Self-attention Layer that computes the relationship between different words in the sequence, as well as a Feed-forward layer. In addition to these, it also has Residual skip connections around both layers along with two LayerNorm layers.

4.3.1 Embedding Layer

Given a sequence of words as inputs, the Embedding Layer has the purpose of converting it into numerical vectors processable by the model. These input repre-

UPDF
WWW.UPDF.COM

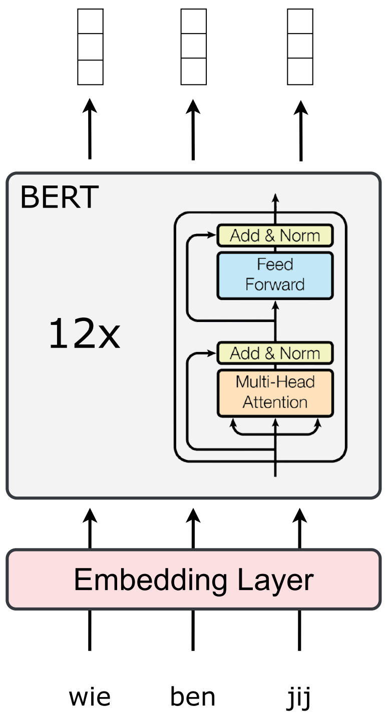


FIGURE 4.6: The Transformer based BERT base architecture with twelve encoder blocks. Input strings are converted into contextualized word embeddings in the output.

sentations are constructed by summing three different types of embedding (FIGURE 4.7):

- Token Embeddings which are pre-trained vectors representations of words
- Segment Embeddings which are vectors to help BERT distinguish between paired input sequences.
- Positional Embeddings which encodes the absolute positions from 1 to maximum sequence length.

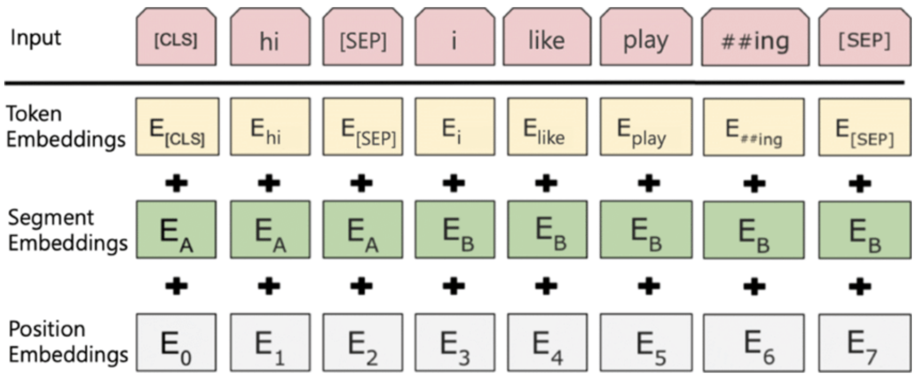

Input
[CLS]
hi
[SEP]
i
like
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{hi} $E_{[SEP]}$ E_{i} E_{like} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7

FIGURE 4.7: BERT Embedding Layer from (Devlin et al., 2018). The input embeddings are the sum of three elements: the Token Embeddings, the Segment Embeddings, and the Position Embeddings.

Firstly, the input sentence is tokenized with WordPiece (Wu et al., 2016), a subword tokenization algorithm. This means that it optimizes the ratio between

UPDF
WWW.UPDF.COM

the number of known words and the total size of the vocabulary decomposing rare words in the so-called word pieces while keeping intact frequently used words. For instance, *"Bayesian"* might be considered a rare word and could be decomposed into "Bayes" and "##ian". Both "Bayes" and "##ian", as independent word pieces, appear more frequently while at the same time the meaning of "Bayesian" is secured by the composite meaning of "Bayes" and "##ian".

Thanks to a vocabulary dictionary that assign each known word with a unique numerical identifier, each token is substituted with Token Embedding. BERT's vocabulary was constructed using Wikipedia articles and contains approximately 30,000 of most common words and subwords found in the English language. All the tokens in the input text are therefore converted into their corresponding identifiers. In the Embedding Layers, each identifier is also associated with a word embedding of size d_{BERT} which replace the identifier in the sequence, as shown in FIGURE 4.8. These vectors are pre-trained representations but during the training they are further optimized for the target task.

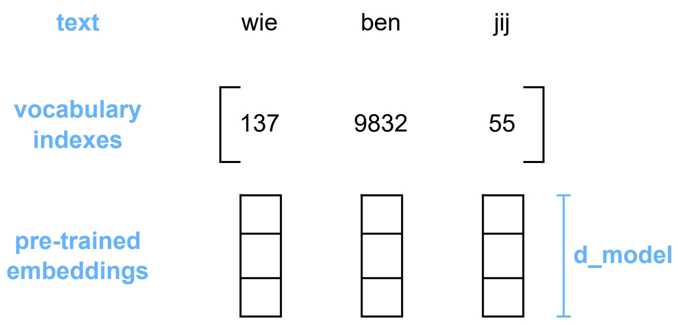


FIGURE 4.8: Conversion from string of text to pre-trained Token Embeddings.

Since the tokens in the input are not processed sequentially but in parallel, it is necessary to inform BERT about the order of the tokens in the sentence. Positional Embeddings encode this information is using wave-frequency functions that depend on the max length of the input sequence. More formally:

$$PE_{pos,2i} = sin\left(\frac{pos}{1000^{\frac{2i}{d_{BERT}}}}\right) \tag{4.2}$$

$$PE_{pos,2i+1} = cos\left(\frac{pos}{1000^{\frac{2i}{d_{BERT}}}}\right) \tag{4.3}$$

where pos is the position of a token in the input sequence, i is the dimension in the embeddings, and d_{BERT} the size of embeddings output by the Embedding Layer.

BERT can solve NLP tasks that involve text classification given a pair of sentences. In such cases, the different input texts are differentiated in two ways. First, a [SEP] token is placed in between them. Second, a learned embedding is added to every token indicating whether it belongs to the first or second sentence. The dimensions of Token Embeddings, Segment Embedding, and Positional Embeddings are illustrated in FIGURE 4.9.

UPDF
WWW.UPDF.COM

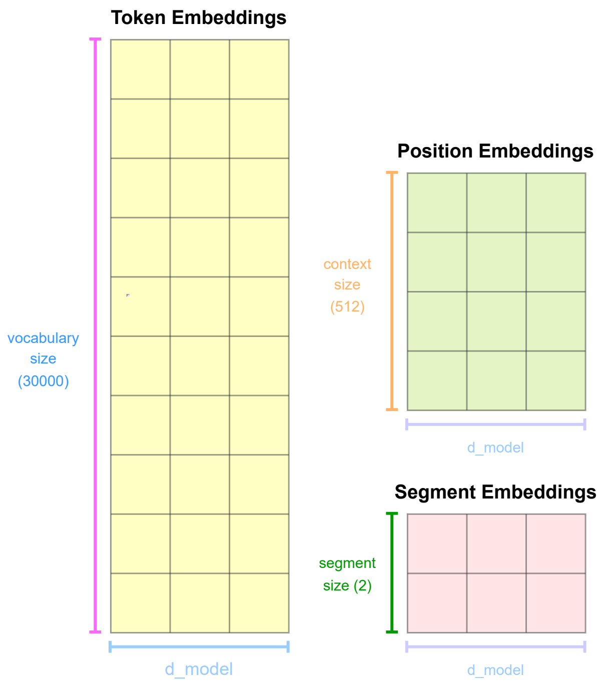


FIGURE 4.9: Parameters of the Embedding Layer.

4.3.2 Encoder Layer

The Encoder Layer takes the embedding created in the previous phase and converts them into dense vector representations that hold the learned information from the entire sequence. It contains two submodules: a multi-headed attention module followed by the fully connected layer, the Residual skip connections, and the LayerNorm layers.

4.3.3 Multi-Headed Attention Module

The multi-headed attention module applies a specific attention mechanism called self-attention that allows the model to correlate each word in the input sequence with the other words in the input. To achieve self-attention, the input embeddings coming from the Embedding Layer are fed to three distinct fully connected layers to create the so-called query vector $\boldsymbol{q}_i$, key vector $\boldsymbol{k}_i$, both of dimension d_k, as well as a value vector $\boldsymbol{v}_i$ of dimension d_v. More formally, for each term $\boldsymbol{x}_i \in \mathbb{R}^{d_{BERT}}$ in an input vector $\boldsymbol{x} = \{x_1, x_2, ..., x_N\}$ of length N, we have:

$$\begin{aligned} \boldsymbol{q}_i &= \boldsymbol{x}_i \boldsymbol{W}^Q \\ \boldsymbol{k}_i &= \boldsymbol{x}_i \boldsymbol{W}^K \\ \boldsymbol{v}_i &= \boldsymbol{x}_i \boldsymbol{W}^V \end{aligned} \tag{4.4}$$

where $\boldsymbol{W}^Q \in \mathbb{R}^{d_{BERT} \times d_k}$, $\boldsymbol{W}^K \in \mathbb{R}^{d_{BERT} \times d_k}$, $\boldsymbol{W}^V \in \mathbb{R}^{d_{BERT} \times d_v}$ are the weight matrices of the fully connected layer used to compute the three vectors, respectively. This step is illustrated in FIGURE 4.10a.

The queries and keys undergo a dot-product matrix multiplication to produce a score vector s. The score vector determines how much focus should word put on

UPDF
WWW.UPDF.COM

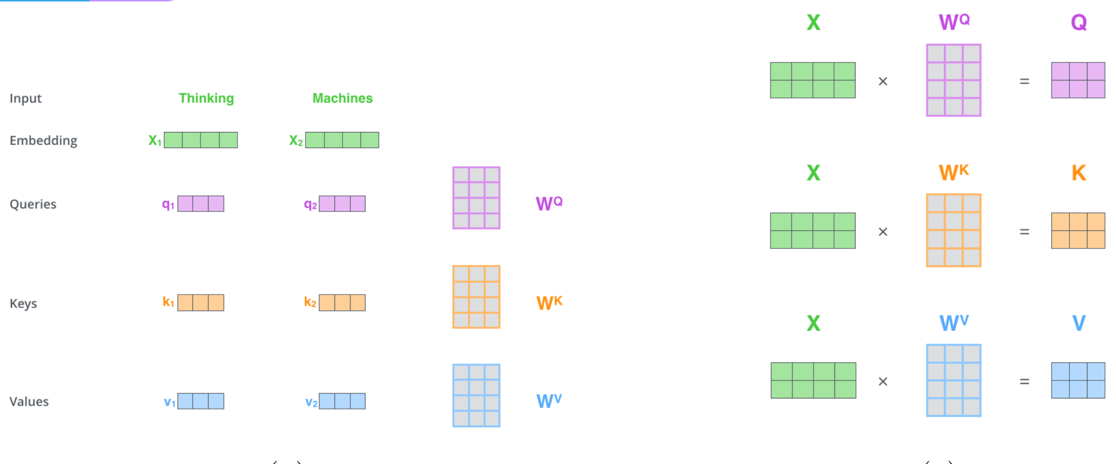


FIGURE 4.10: Computation of the query $\boldsymbol{q}$, key $\boldsymbol{k}$ and value $\boldsymbol{v}$ vectors (A). Computation of the query $\boldsymbol{Q}$, key $\boldsymbol{K}$ and value $\boldsymbol{V}$ matrices (B). Illustration from (Alammar, 2018).

each other. The higher the score the more attention is given to that word.

$$s_{ij} = \boldsymbol{q}_i * \boldsymbol{k}_j \tag{4.5}$$

Hence, the attention weights are calculated by applying a softmax function to the score matrix divided by d_k, the dimension of the queries and key vectors.

$$s'_{ij} = softmax\left(\frac{s_{ij}}{\sqrt{d_k}}\right) v_j \tag{4.6}$$

The obtained probabilities are multiplied by the values vector. Here, important words are valorized by high probabilities while irrelevant ones are given less attention.

In practice, the self-attention function is computed on a set of queries simultaneously, packed together into a matrix $\boldsymbol{Q} \in \mathbb{R}^{N \times d_k}$. As shown in FIGURE 4.10b, the keys and values are also combined together into respective matrices $\boldsymbol{K} \in \mathbb{R}^{N \times d_k}$ and $\boldsymbol{V} \in \mathbb{R}^{N \times d_v}$. The output of a single head is therefore computed as follow:

$$Attention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = softmax\left(\frac{\boldsymbol{Q}\boldsymbol{K}^T}{\sqrt{d_k}}\right) \boldsymbol{K} \tag{4.7}$$

As illustrated in FIGURE 4.11a, this operation is repeated in each of the attention heads in the multi-headed attention module, linearly projecting the queries, keys and values h times with different learned projections. The result is further projected into another representation subspace with a matrix $\boldsymbol{W}^O \in R^{d_{BERT} \times d_{BERT}}$, such that:

$$MultiHeadAttention(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = Concat(head_1, head_2, ..., head_h)\boldsymbol{W}^O$$
$$\text{where } \; head_i = Attention(\boldsymbol{Q}\boldsymbol{W}_i^Q, \boldsymbol{K}\boldsymbol{W}_i^K, \boldsymbol{V}\boldsymbol{W}_i^V)$$

here the projections are parameter matrices $\boldsymbol{W}_i^Q \in \mathbb{R}^{d_{BERT} \times d_k}$, $\boldsymbol{W}_i^K \in \mathbb{R}^{d_{BERT} \times d_k}$, $\boldsymbol{W}_i^V \in \mathbb{R}^{d_{BERT} \times d_v}$ for each $head_i$. This is done to add variety to the attention mechanism: each head is supposed to capture different patterns in the input sequence,

UPDF
WWW.UPDF.COM

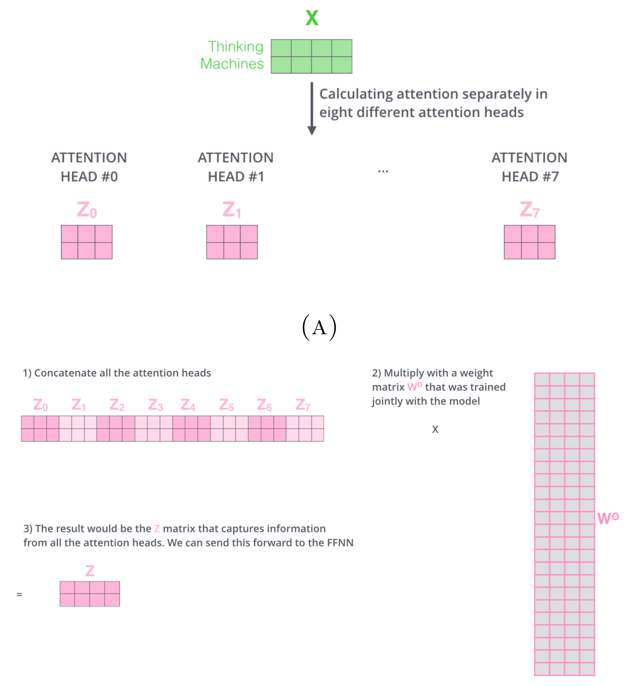

X
Thinking
Machines
Calculating attention separately in eight different attention heads
ATTENTION HEAD #0
ATTENTION HEAD #1
...
ATTENTION HEAD #7
Z_0
Z_1
Z_7
(A)
1) Concatenate all the attention heads
Z_0 Z_1 Z_2 Z_3 Z_4 Z_5 Z_6 Z_7
2) Multiply with a weight matrix W^O that was trained jointly with the model
x
W^O
3) The result would be the Z matrix that captures information from all the attention heads. We can send this forward to the FFNN
Z
=
(B)

FIGURE 4.11: Illustration of the multi-head attention mechanism (Alammar, 2018). In (A), the computation of the attention heads. In (B), the computation of the final multi-head attention output

producing diverse attention scores, and giving the encoder model more representation power. The output produced by each head is then concatenated before being fed to the linear layer (see FIGURE 4.11b).

The Transformer architecture uses self-attention by relating every word in the input sequence to every other word. Considering the sentences:

- The *cat* drank the milk because *it* was hungry.
- The cat drank the *milk* because *it* was sweet.

In the first, the word ‘it’ refers to ‘cat’, while in the second it refers to ‘milk. When the model processes the word ‘it’, self-attention gives the model more contextual information about its meaning so that it can associate ‘it’ with the correct word. This relations will be captured in the features of the generated word embeddings.

4.3.4 Residual Connection, Layer Normalization and Point-wise Feed Forward

For each of the layers in the encoder, the multi-headed attention output is added to the original input through the Residual Connection and then passed through a Layer Normalization. The result is fed to a point-wise feed-forward (FFN) network which consists of two linear layers with a ReLu activation in between. More specifically:

$$FFN(\boldsymbol{x}) = max(0, \boldsymbol{x}\boldsymbol{W}_1 + \boldsymbol{b}_1)\boldsymbol{W}_2 + \boldsymbol{b}_2 \tag{4.8}$$

While the linear transformations are the same across different positions, the parameters $\boldsymbol{W}_1 \in \mathbb{R}^{d_{BERT}*d_{ff}}$, $\boldsymbol{b}_1 \in \mathbb{R}^{d_{BERT}}$, $\boldsymbol{W}_2 \in \mathbb{R}^{d_{BERT}*d_{ff}}$, and $\boldsymbol{b}_2 \in R^{d_{BERT}}$ are

UPDF
WWW.UPDF.COM

different from layer to layer. dff represents the dimensionality of the inner-layer and is generally set to $dff = 4 * d_{BERT}$.

The output is again added to the initial input and further normalized.

$$LayerNorm(\boldsymbol{x} + FFN(\boldsymbol{x})) \tag{4.9}$$

The residual connections help the network train by allowing gradients to flow through the network directly. The Layer Normalization has the function to stabilize the gradients and enables faster training. Lastly, the point-wise FFN is used to process the attention output to gain a reacher representation.

4.4 Pre-Training

The BERT model has been applied to many NLP tasks, such as Question Answering, Sentiment Analysis, Text summarization, and so forth. All of these problems require a basic understanding of language. This is where the pre-training phase comes in. Here, the model is taught about language rules and context through two unsupervised tasks performed simultaneously: Masked Language Model (MLM) and Next Sentence Prediction (NSP).

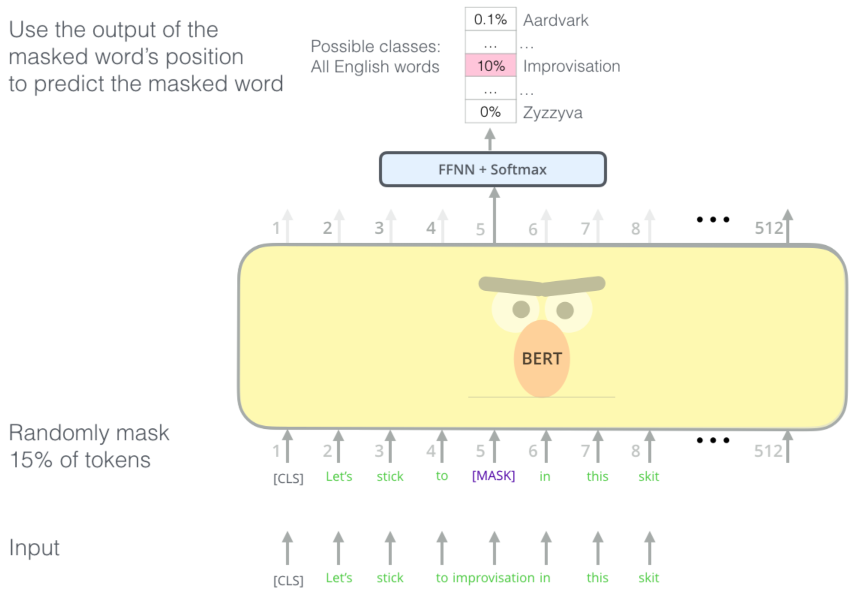

Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
...
10% Improvisation
...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

FIGURE 4.12: Illustration of the MLM task from (Alammar, 2019)

In MLM (FIGURE 4.12), BERT receives a masked sentence in which some words are omitted and substituted by the special token [MASK]. More specifically, in 15% of cases, a random word in the sequence can be replaced by [MASK] (80% of the time), by a random word (10% of the time), or by the same word (10% of the time). The objective is to predict a probability distribution over the known vocabulary to match the substituted word. This helps BERT understand bidirectional context within a sentence.

In NSP (FIGURE 4.13) the input are combined pairs of sentences where the special [CLS] token is placed at the beginning and the [SEP] token at the end of both the sentences. Once the input passes through the model, the output vector of the [CLS]

UPDF
WWW.UPDF.COM

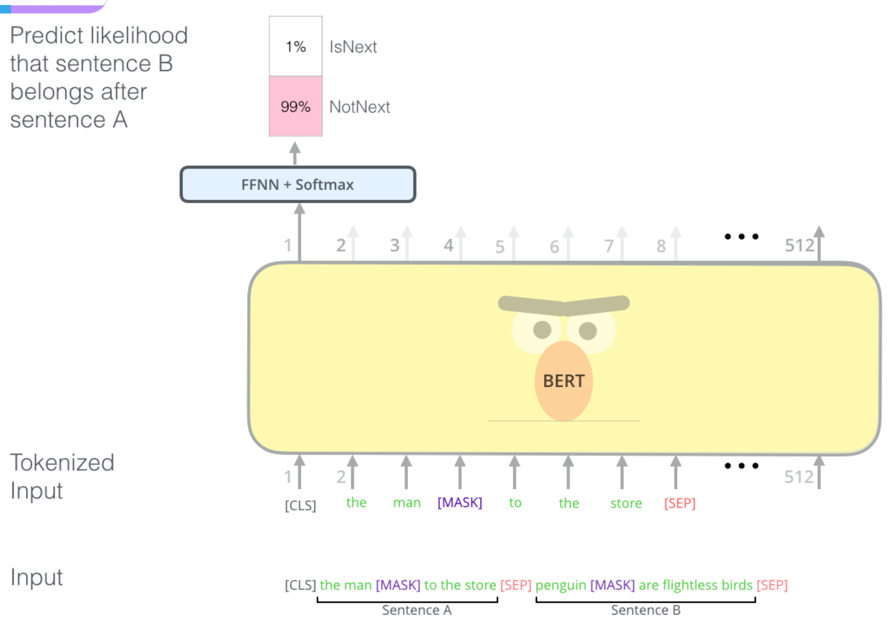

Predict likelihood that sentence B belongs after sentence A
1% IsNext
99% NotNext
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Tokenized Input
[CLS] the man [MASK] to the store [SEP]
Input
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]
Sentence A
Sentence B

FIGURE 4.13: Illustration of the NSP task from (Alammar, 2019)

token is taken as the representation of the whole sequence. This is then fed to a linear classifier that predicts IsNext or NotNext depending if the sentences are part of the same text or just a random combination from different texts. This informs BERT about context between sentences.

Given the unsupervised nature of this phase and the efficiency of such a model in processing text data, it has been possible to intensively train BERT on large amounts of data. More specifically, the authors of BERT used the BooksCorpus (800M words) (Zhu et al., 2015) and the text passages extracted from the entire English Wikipedia (2,500M words). The result is that the Transformer encoder learns about natural language while "reading" natural language and in the meanwhile, it implicitly acquires what practitioners call a basic understanding of language, together with its syntactical and semantical rules.

4.5 Fine-Tuning: Downstream Tasks

At this point, BERT can be further trained using a supervised approach on the so-called downstream tasks. As illustrated in FIGURE 4.14, These can be divided into four categories: (1) sentence pairs classification tasks (e.g. textual entailment and semantic similarity) (2) single sentence classification tasks (e.g. sentiment analysis and topic detection), (3) question-answerings tasks (e.g. open-domain QA), and (4) single sentence tagging tasks (e.g. POS tagging and NER). For each task, a final fully connected output layer is added on top of BERT. Then the task-specific input is simply plugged into the model whose parameters are fine-tuned end-to-end.

This second training phase is less intensive compared to the pre-trained step. Given that the parameters already memorized a broad range of linguistic features, fine-tuning results in a much faster process that requires fewer data and less computational power to be completed.

During fine-tuning, the model parameters adapt to solve a specific task develop-

UPDF
WWW.UPDF.COM

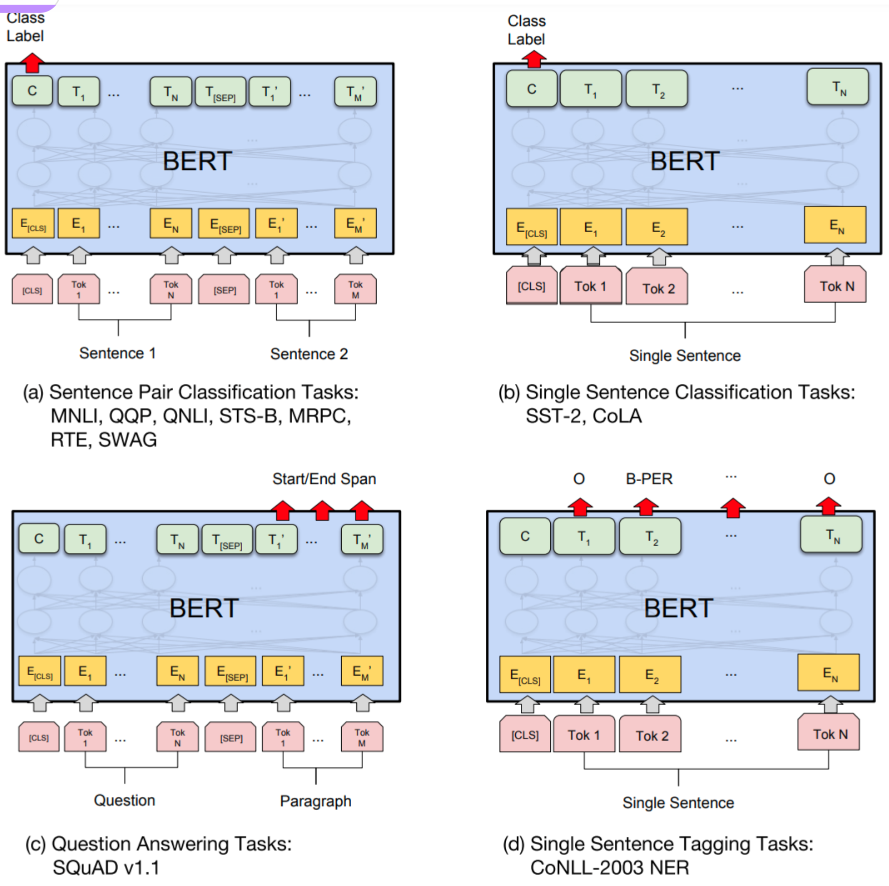


Figure 4.14: Illustration from (Devlin et al., 2018) that shows four ways to use BERT for different downstream tasks.

ing more specialized features as these are the ones that need to be repurposed on the new problem. Hence, the importance of the datasets whose purpose is present the model with clear examples to correctly condition such features for the target task.

UPDF
WWW.UPDF.COM

Chapter 5

Dataset Construction

All supervised machine learning models are built on a general axiom: map an input to an output based on patterns during the training phase. Through this learning paradigm, many complex tasks, such as regulatory genomics, financial market prediction, control problems, protein folding, etc., found solutions with high levels of accuracy. In this context, datasets constitute the backbone for supervised learning research since necessary to train, evaluate and compare machine learning models. Given their importance to the field, the construction of machine learning corpora has been itself object of research (Gebru et al., 2021). While the purpose of many datasets is to train new models using big amounts of data, others focus on probing tasks that challenge existing algorithms on a specific pattern recognition problem (Paperno et al., 2016), (Poerner, Waltinger, and Schütze, 2019).

Despite the contribution that they bring to the scientific community, datasets are a mixed blessing. The dependence of deep models on training corpora leads researchers' efforts to gravitate around the problems these benchmarks address, fostering the *SOTA-chasing effect* (Church and Kordoni, 2022) where better performance goes hand-in-hand with poorer insights, and ultimately hindering the diversification in the research. A recent study conducted in collaboration between the University of California and Google™ (Koch et al., 2021) examined causes and impact of this trend. The research reveals a concentration on fewer datasets within task communities. In response to this, transfer learning has gained more popularity in different research groups, especially in Computer Vision and in the broad body of tasks belonging to the "Methodology" category in the Papers With Code (PWC) corpus[1]. Although the NLP scientific community seems to be the least affected by this phenomenon, the researchers also illustrate how over 50% of dataset usage in PWC can be attributed to just a dozen institutions that alone have a great impact on the agendas of machine learning research.

Research diversification is particularly challenging when operating in a domain-specific scenario. From machine translation to knowledge extraction, domain experts are necessary for high-quality annotations. Reason for which the production of such corpora is costly in terms of time and funds.

Being directly affected by these circumstances, I propose an alternative approach to manual annotation. The method allows the creation of domain-specific datasets for training AKE supervised models. With minimal fine-tuning of a few hyperparameters, the pipeline annotates a collection of academic textbooks, highlighting

[1]https://paperswithcode.com/

UPDF
WWW.UPDF.COM

meaningful terms that can be used in various evaluation tasks, e.g. document similarity, keyword identification, document clustering, etc. The system makes use of a data structure that I named Global Corpus (GC) to identify the position of index entries in the body of the books. More details on the construction of the GC and the annotation procedure in §5.2.

5.1 Dataset Specifics

giving example of the dataset is common practuce

This section gives a brief description of the major characteristics of the corpus used to train the AKE algorithm. It includes 9 textbooks focusing on the Statistics domain: [1] *A Concise Guide to Statistics* (Kaltenbach, 2007), [2] *A Modern Introduction to Probability and Statistics* (Dekking et al., 2005), [3] *Modern Mathematical Statistics with Applications* (Jay L. Devore, 2014), [4] *OpenIntro statistics* (Diez, Barr, and Cetinkaya-Rundel, 2012), [5] *Statistics and Probability Theory* (Faber, 2012), [6] *Statistics for Non-Statisticians* (Madsen, 2016), [7] *Probability and statistics for engineers and scientists* (Walpole et al., 1993), [8] *Statistics for scientists and engineers* (Shanmugam and Chattamvelli, 2015), [9] *Introductory statistics with R* (Dalgaard, 2020). Each raw in the dataset corresponds to a single paragraph from a book, associated with the index entries found in that span of text, and the chapter heading to which the passage belongs. TABLE 5.1 illustrates a snippet taken from the corpus. The highlighted words are the instances of the index terms that occur in the corresponding paragraph. They represent the keywords that the model is supposed to recognize.

Chapter heading: Bayesian Definition **Paragraph:** The degree of belief is also referred to as a prior belief or prior probability, i.e. the belief, which may be assigned prior to obtaining any further knowledge. It is interesting to note that Immanuel Kant developed the philosophical basis for the treatment of subjectivity at the same time as Thomas Bayes developed the mathematical framework later known as bayesian statistics.	**Keywords:** bayes, bayesian statistics, prior belief, prior probability

TABLE 5.1: An exemplar raw from the generated dataset. The terms highlighted are the keywords the AKE algorithm is asked to retrieve.

The division into paragraphs comes from two main considerations. First, a paragraph is the smallest discursive unit in a book that argues about a single idea. Indeed, pages or book chapters are too broad and may cover more than one idea. Conversely, sentences are too short and do not provide enough context. Second, the BERT-based model that was trained on the statistical corpus accepts text sequences up to a defined limit which forbids inputting entire pages or books. Given this constriction, the division into paragraphs seemed the best trade-off between input length and passage informativeness.

WWW.UPDF.COM

The general metrics of the dataset are summarized in Table 5.2. All the values refer to the are corpus obtained after the preprocessing phase described in §5.3.2 and §5.3.1.

Books	#Words	#Paragraphs	#Words per paragraph m (sd)	Index terms	%Keywords	#Keywords per paragraph m (sd)
[1]	39567	697	57 (43)	101	10.7%	6 (5)
[2]	129441	3151	41 (39)	178	11.1%	5 (5)
[3]	327219	5127	64 (89)	312	7.8%	5 (7)
[4]	110761	2854	39 (32)	225	11.6%	4 (4)
[5]	46577	1008	46 (37)	178	12.6%	5 (5)
[6]	45197	1236	36 (24)	73	10.6%	3 (3)
[7]	212970	4409	48 (47)	221	10.8%	5 (5)
[8]	186125	3934	47 (52)	223	8.4%	3 (4)
[9]	83767	1844	45 (30)	128	9.2%	4 (4)

TABLE 5.2: A tabular summary of the textbooks included in the dataset.

Depending on the task, different strategies can be adopted to split the dataset for training and inference. It is possible to opt for a standard 80/10/10 division in train, test, and validation sets. However, given the particular nature of textbooks, in which the specific subject can vary significantly even within the same domain, it is advisable to maintain each book intact and divide them to have an optimal ratio between the sets. For the experiment described in §7.1, I decided to use a cross-validation approach. Therefore, the model was trained on eight books and tested on one. This operation was then repeated for as many iterations as the number of documents in the corpus. In each iteration, a different textbook was included in the test set.

in scintific articles practical advice are not necessary: it is supposed that who reads these paper already have a strong backgoring knowledge in ML

Data imbalances pose a challenge for classification models as most of the machine learning algorithms assume an equal number of examples for each class. Showing the model with fewer examples results in poorer predictive performance, specifically for the minority class which in many cases is also the most important one. Therefore, before using a dataset, it is important to be aware of possible asymmetries. FIGURE 5.1 displays two different class imbalances to take into account if training a model on this corpus. FIGURE 5.1a shows the distribution of shared index entries *across* documents. Noticeable is the fact that more than 60% of the terms are solely present in one textbook; meaning that the vast majority of the other terms will be difficult to retrieve due to the low number of books containing them. FIGURE 5.1b, on the other hand, presents the distribution of index terms *within* the whole corpus. Together with a sparsity of terms across textbooks, there is a long tail of terms with a low occurrence frequency. It is reasonable to think that these two distributions share a strong correlation: the most common terms across textbooks are likely to abound the total distribution in the whole dataset, and vice versa.

UPDF
WWW.UPDF.COM

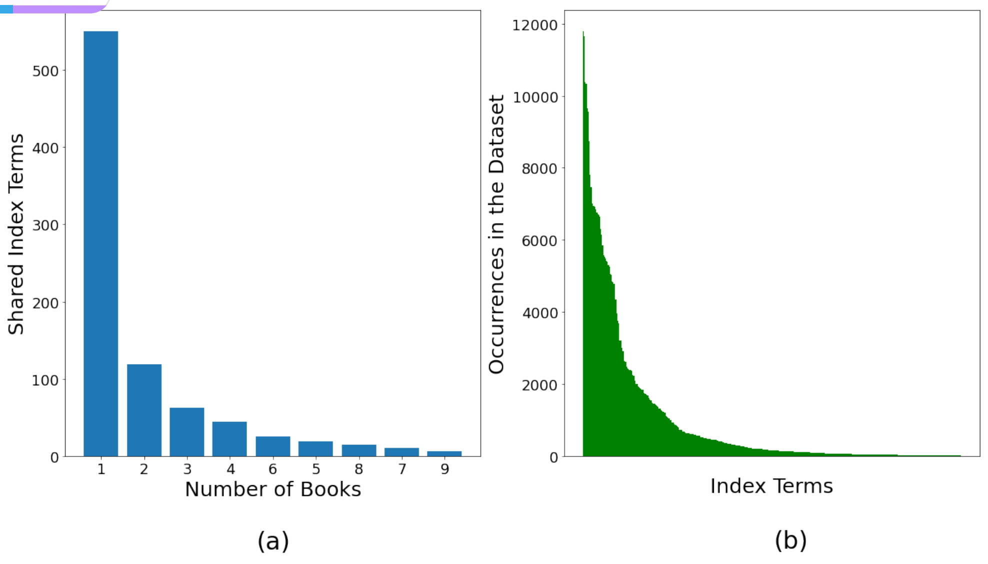


FIGURE 5.1: Two imbalances in the dataset. (A) shows the distribution of index terms *across* the nine books that constitute the corpus. Each column represents how many index terms are shared by a number of books equal to the value in the abscissa. The graph illustrates how the majority of terms are solely present in one document. (B) describes the distribution of index terms *within* the corpus. The abscissa represents each index term included in the various indexes while the ordinate is the corresponding total count of occurrences in the dataset.

5.2 Dataset Annotation: Global Corpus

The annotation of the dataset is performed by identifying all the instances of the index terms in the body of the books included in the corpus. An index term is not mapped solely to the paragraphs of the book where it belongs, but its instances are also identified in all the other documents, regardless of whether it appears in the corresponding indexes. This ensures coherence across the textbooks.

The index terms are tabulated in a common repository previously referred to as GC. The purpose of the GC is not solely to contain the index terms but also to associate them with additional information. Each index entry is here paired with some ”properties” derived from the Domain Knowledge (§3.2) of the textbooks. The properties are the following: (i) the domain specificity, i.e. *core-domain*, *in-domain*, *related-domain*, or *out-of-domain*; (ii) an attribute specifying if the term is a *main heading* or *subheading* depending on how it was inserted in the index; (iii) a list of synonyms; (iv) the books in which at least an occurrence of the index term is recognized. The list of synonyms is obtained from various sources. First, the books used in the dataset: thanks to the Knowledge Base of each document, the authors of (Alpizar-Chacon and Sosnovsky, 2020) identified all the wording of an index term in the different textbooks and group them. Additionally, they included the synonyms found in the ISI Glossary of Statistics[2]. Second, if an index term is associated with a DBpedia link, the title of the corresponding webpage is added.

This accounts for different choices of terminology that book authors decide to follow. For example, the terms ”mean” and ”average”, although different morphologically, represent the same concept, i.e. the sum of the values of the items in a

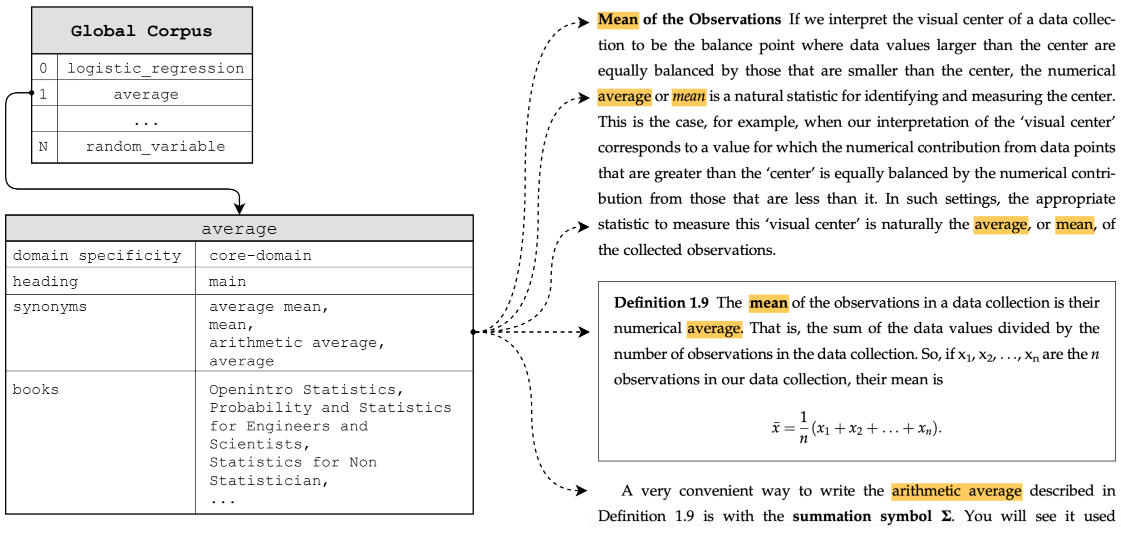
[2]https://www.isi-web.org/isi.cbs.nl/glossary/

UPDF
WWW.UPDF.COM

group divided by the number of items. In such cases, the GC joins the two terms.

FIGURE 7.2 illustrates how the GC is structured and how the list of synonyms is used to label the paragraphs in the corpus. Once the paragraphs have been lowercased and lemmarized, the annotation works with an exact match comparison between the passages and the terms included in `synonyms`.

However, before starting to annotate the textbooks, the GC and all the paragraphs went through a series of filters to clean the corpus from irrelevant or dirty data. These filters are closely described in the next sections.



Global Corpus	
0	logistic_regression
1	average
	...
N	random_variable

average	
domain specificity	core-domain
heading	main
synonyms	average mean, mean, arithmetic average, average
books	Openintro Statistics, Probability and Statistics for Engineers and Scientists, Statistics for Non Statistician, ...

Mean of the Observations If we interpret the visual center of a data collection to be the balance point where data values larger than the center are equally balanced by those that are smaller than the center, the numerical average or *mean* is a natural statistic for identifying and measuring the center. This is the case, for example, when our interpretation of the 'visual center' corresponds to a value for which the numerical contribution from data points that are greater than the 'center' is equally balanced by the numerical contribution from those that are less than it. In such settings, the appropriate statistic to measure this 'visual center' is naturally the average, or mean, of the collected observations.

Definition 1.9 The **mean** of the observations in a data collection is their numerical average. That is, the sum of the data values divided by the number of observations in the data collection. So, if $x_1, x_2, \ldots, x_n$ are the n observations in our data collection, their mean is

$$\bar{x} = \frac{1}{n}(x_1 + x_2 + \ldots + x_n).$$

A very convenient way to write the arithmetic average described in Definition 1.9 is with the **summation symbol Σ**. You will see it used

FIGURE 5.2: On the left the GC, containing all the index terms retrieved from the textbooks. Each index term is associated with four properties. In particular, the list of synonyms is used to annotate the body of the books. On the right, a snippet from *Intuitive Introductory Statistics*, with the corresponding highlighted keywords derived from the list `synonyms` of the index term term `average`.

5.3 Dataset Pre-processing

The production of a dataset from scratch presents various levels of challenge, e.g. sources validation, data collection, data processing, etc.; all of them critical for the final result. Particularly in the field of Artificial Intelligence, the pre-processing of data is of primary importance to produce deep models that are both performant in the goal task and robust to unseen data. *"Garbage In Garbage Out"* (GIGO) is a notorious expression among ML practitioners and it clearly expresses that the state of the input directly influences the quality and the reliability of the output (Garrido-Muñoz et al., 2021).

Another element which is influenced by the quality of the dataset is the model behavior. Despite the great ability of ML systems to unfold patterns in a large amount of data, their obscure structure poses a limit to explaining their internal functioning. The generation of a good quality dataset helps mitigate this unpredictability, under the assumption that the cases shown during the training phase will continue to unfold in the future. Properly pre-processing the data means giving a clearer image of what the model should look at and reducing anomalous behaviors

UPDF
WWW.UPDF.COM

during inference. The identification of such ”garbage” is the first step to having accurate predictions since even the best model is doomed to perform poorly if trained on a low-quality dataset. In short, pre-processing is a necessary step to follow to avoid low performances, unforeseen model behaviors, or bias in the decision-making phase.

Given this premise, a main concern was to prepare a corpus keeping in mind the concept of GIGO. Therefore, a pre-processing step rules out noise entities that would negatively affect the behavior of the models. The following section describes how the task is approached. The first part defines why some index terms were kept or discarded and the reasons behind these choices. Hence, a second step focuses on maintaining a high quality of the passages extracted from the books of the corpus.

5.3.1 Index Terms Filtering

The construction of the term bank is based on the PDF parser described in §3.2. The authors that developed this system used it to generate VSMs for academic textbooks. In the research, all the index entries were used.

Open discussions with professional indexers, however, disclosed the fact that the nature of index terms and their relatedness to the target domain is diverse. Hence, to give the model a clearer image of what kind of terms should be extracted, these differences are to be taken into account. To do this, it is necessary to have a classification describing the various kind of index entries. However, establishing such formal categories is a challenging task due to the particular feature of each individual book index. Although indexers are called to respect the guidelines expressed in the ISO999:1996[3], the general formatting and the overall quality of an index can vary significantly, making it difficult to create general rules that apply to all cases. This and the personal approaches adopted by indexers cause inconsistencies across indexes.

To better understand how indexes are compiled and the reasons behind these discrepancies, diverse indexers were involved in the project to gain more insight into the procedure of compiling an index. In particular, one of these conversations (M. Gee, personal communication, June 29, 2022) clarified what terms are more likely indexed. Thanks to this exchange of emails, I produced a classification of index terms in 3 groups depending on the nature of the referenced terms:

- *factual entries*: this is the major category since most of the entries fall in this group. All these index terms concern facts and notions presented in the book. They may be proper names of individuals, groups of people, organizations, places, or products, concrete objects, processes or activities, techniques, opinions, abstract ideas, theories and models, events, qualities of a person or object.

- *non-textual entries*: these terms refer to embedded elements in the book that differ from pure textual information. Some of these elements are tables, graphs, illustrations, photographs, algorithms and pseudo-code fragments, diagrams, formulas, etc.

[3]https://www.iso.org/standard/5446.html

UPDF
WWW.UPDF.COM

- *didactic entries*: these entries come as a consequence of the nature of textbooks is not only notional but also didactical and therefore intended for instruction. Elements like case studies, examples, exercises and information about additional learning resources do not carry a particular meaning per se but they are useful didactic tools that authors use to better explain in plain terms a complex concept. The key idea here is that didactic entries serve as a memorandum to readers that might want to look up a specific explanatory section of the book. The "blu-cherry tree exercise, 5, 10" index term is a case in point. Many terms non strictly related to the general topic of the document fall in this category as they are often used in examples or exercises.

As indicated in §3.1, it is possible to distinguish index entries in two categories:

- *main headings*: they represent the primary access point used by readers to orient in the index and to rapidly identify the location of interesting terms.

- *sub-headings*: these types of entries are used to specify a determining aspect of the corresponding main heading.

3 assumptions

Given the classification and considering the variability of index entries, this research makes three necessary assumptions: (1) only factual entries that are relevant to the target domain are considered; (2) the entries indicated as *out-of-domain* by the value of `domain specificity` (see FIGURE 7.2) are discarded; and (3) subheadings are included only if they present a domain class.

Factual entries were automatically considered to be domain-related if classified as *core-domain*, *in-domain*, and *related-domain* in the `domain specificity`. However, not all the entries were associated with a category. For unclassified index terms, it was necessary to manually check them to determine which ones were related to the target domain. This procedure was conducted trying to respect as closely as possible the approach adopted by professional indexers. As reported in an interview, by one of these experts (full transcripts are presented in Appendix A), there is not an official resource that they refer to when indexing a book. Therefore, following their suggestions, I relied on a diversified group of sources of information, i.e. Wikipedia web pages, ISI glossary, and other statistical textbooks, to conclude if a term was meaningful or not for the target domain.

Subheadings devoid of domain specificity class were discarded for different reasons. First, the rate of unclassified index subentries significantly was higher compared to main headings. This is a consequence of the fact that subheadings often represent more specific concepts that are challenging to recognize and classify (see (Alpizar-Chacon and Sosnovsky, 2022) for more details on how index entries are categorized). Leaving such entries would have meant significantly increasing the number of index terms to manually check. The presence of a domain class and the second condition ensure that these index terms are related to the target domain. Second, the synonyms retrieved by the PDF parser were not always precise, especially in the case of subentries. For example, the index term "independence+chi-squared_test_for" (the + splits the sub heading, on the right, by its corresponding main heading) was associated with the sentence "chi-squared test of example 13.13 testing for independence". Labeling the dataset with such phrases would have contained the annotation with irrelevant words.

UPDF
WWW.UPDF.COM

3 assumptions + 1

In addition to the three assumptions, ambiguous entries were also filtered. On occasions, index terms lacking of a proper modifier may be associable with more domains if taken out of context. An example is the word "process". A process is quite a generic term that can be found in statistical textbooks, e.g. in "stochastic process", as well as in documents not necessarily related to Statistics. From biochemistry, "apoptosis process" is an example. In the case an index term was considered to be ambiguous, the entry was excluded from the GC.

All the above-described filters aim to maintain a topical coherence among the terms included in the term bank. Reducing the number of index entries allows for better controllability in the dataset and, consequently, more interpretability of the model results.

synonyms filtering

As already said, the corpus is labeled using the `synonyms` of each index term. Although a great deal of contaminated data was left out by the previous step, some were still imprecise: occasionally, synonyms only partially match the corresponding index term. For example, it was found that the index term "alternative hypotheses" included the instance "hypothesis". Although "hypotheses" is indeed a statistical term, "alternative hypotheses" comes with a more specific meaning indicating one of the two proposed propositions in hypothesis testing. For this reason, all the `synonyms` of the index terms that passed the first filtering phase were also manually checked; those being partial matches were promptly removed from the term bank.

5.3.2 Textbook Passage Filtering

paragraph filetring

PDF Parsers do not always capture all the elements in a document. Graphs, images, tables, or mathematical formulas are often not correctly recognized and therefore generate noise sequences of characters. To mitigate this effect and maintain a consistent quality across the paragraphs, the following four filters are applied:

- *paragraph-length*: paragraph lengths ($|S|$) shorter than a threshold α are discarded to maintain a good quality of contextualized word embedding.

$$|S| > \alpha$$

- *ratio-characterize-words*: checks the sparseness of the string. If there are many single characters the value will overcome the threshold. It is reasonable to think that in a sentence the number of single characters ($|s|$) is greater than the number of words. The string "p(a ∪ b) = p(b) + p(a ∩ bc)" is an example.

$$\frac{|S|}{|s|} < \beta$$

- *ratio-of-digits*: is the number of digits normalized per sequence length. The number of digits may differ depending on the goal domain. However, noise strings can be identified for their anomalous number of digits compared to the total number of characters in the string.

$$\frac{|s \textit{ iif } s \in \mathcal{N}|}{|s|} < \gamma$$

UPDF
WWW.UPDF.COM

- *ratio-of-special-characters*: special characters such as æ, ©, or μ can be easily identified using their Unicode. The number of such character should be limited to few instances in relation to the total number of characters in the sentence. The formula "2π-($\sqrt{50}$y-$\sqrt{32}$x)" is an example.

$$\frac{|s \ iif \ Unicode(s) >= 128|}{|s|} < \delta$$

Depending on the focus of the textbook, these four parameters may vary significantly. For example, in the specific case of Statistics, digits and special characters (e.g. greek letters or mathematical operators) are more frequent than in other domains. On the contrary, in History books the latter is rarely found but digits may still be consistently present in the form of dates. In this case study, I opted for the following values: α, β, γ, δ = {3, 0.30, 0.40, 0.05}

UPDF
WWW.UPDF.COM

Chapter 6

Keyword Extraction Pipeline

6.1 Workflow and Experimental Setting

To ensure the reproducibility of the conducted experiments, this chapter starts with a detailed description of the model architecture and hyperparameters. In the first step of the proposed AKE architecture, textbooks are fed paragraph-by-paragraph into BERT. Then, the encoder generates word embeddings that capture the meaning of each token contextualized to the sentence in which occurs. Subsequently, the embeddings go into a BiLSTM that further processes the information in the vectors, supporting BERT in the classification task. Next up is a dropout layer (Srivastava et al., 2014) which during training randomly omits a number of hidden layer outputs with a probability of 0.5. The noise introduced makes the model less prone to overfit the test set since at each iteration the predictions are generated with a ”new” configuration of hidden layers. Finally, the vector representations pass through a linear classifier that maps each token to two labels, i.e. *keyword* or *non-keyword*. The predictions are then enhanced using a series of syntactical filters to better reflect the index entries produced by professional indexers.

If multiple words are positively classified in sequence, they are aggregated to form a unique keyword. The model was trained for 20 epochs using a binary crossentropy loss function with the ADAM (Kingma and Ba, 2014) optimizer, a first-order stochastic gradient descent algorithm, and a learning rate set to 0.0001. The dataset was input with a batch size of 32, as suggested by the authors of BERT. The test set includes only one textbook while the rest of the documents form the training set. Both of them were shuffled to improve the overall performance and at the same time maintain consistency across different runs.

Due to system memory constraints, DistillBERT is used in place of the original model proposed by Google. DistillBERT, first published by (Sanh et al., 2019), represents a lighter but more efficient version of BERT. Even though it has half of the layers, it runs 40% faster while achieving over 95% of BERT's performance on the GLUE benchmark (A. Wang et al., 2018). More specifically, the pre-trained version publicly available on the HuggingFace [1] platform was used. All the hyperparameters, i.e. embedding size (d_{BERT}=768), number of attention heads (d_{att}=12), number of hidden layer (d_{layer}=12), and size of the intermediate layer (d_{inter}=3072), follow the original set up of the model. The BiLSTM is built from the PyTorch[2] implemen-

[1] https://huggingface.co/
[2] https://pytorch.org/docs/stable/generated/torch.nn.LSTM.html

UPDF
WWW.UPDF.COM

tation and the number of corresponding hidden layers is set to d_{BiLSTM}=125. Also here, a dropout is included but with a 0.05% probability of omission. An overview of the full architecture is illustarted in FIGURE 6.1.

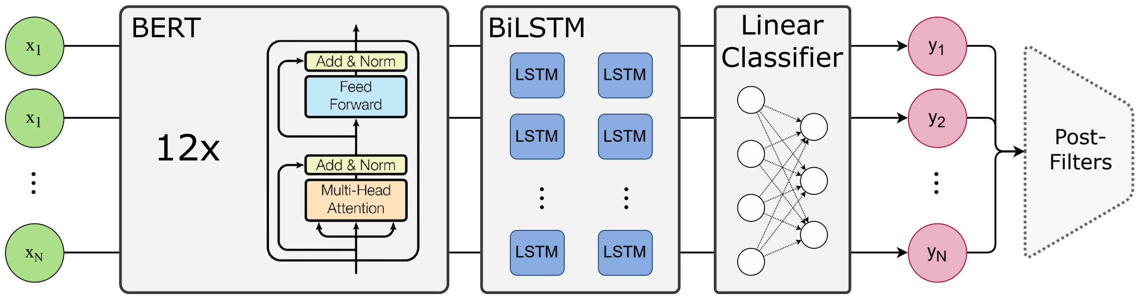


FIGURE 6.1: An overview of the architecture used for the AKE task. The input text is fed to a (1) BERT encoder that generated contextualized word embeddings. These are further processed by a (2) BiLSTM. Lastly, the vector representation passes through a (3) linear classifier that computes a score representing the relevance of the initial word for the target domain. The first three components can be used for standard sequence labeling tasks. In addition, a (4) top layer filters the prediction according to syntactical rules.

6.2 Problem Statement

Suppose a book is defined by a sequence of n text units $\mathcal{U}^{(i)}$, such that $\mathcal{B} = \{\mathcal{U}^{(1)}, ..., \mathcal{U}^{(n)}\}$, where each unit is initially identifiable as a paragraph and the keywords to be identified in that span of text. Therefore, $\mathcal{U}^{(i)}$ contains a set of m tuples:

$$\mathcal{U}^{(i)} = \{(\mathcal{X}^{(1)}, \mathcal{Y}^{(1)}), ..., (\mathcal{X}^{(m)}, \mathcal{Y}^{(m)}\} \tag{6.1}$$

where $\mathcal{X}^{(i)}$ is a list of words and $\mathcal{Y}^{(i)}$ is a list of labels. Both $\mathcal{X}^{(i)}$ and $\mathcal{Y}^{(i)}$ are sequences of size L, such that:

$$\mathcal{X}^{(i)} = x_1^{(i)}, x_2^{(i)}, ..., x_L^{(i)} \tag{6.2}$$

$$\mathcal{Y}^{(i)} = y_1^{(i)}, y_2^{(i)}, ..., y_L^{(i)} \tag{6.3}$$

The final goal is to obtain a model able to consistently map $\mathcal{X}$ onto the tag set $\mathcal{Y} \in \{1, 0\}$, depending if the single word is or not a keyword. It is worth noting that given the particular case of textbooks, not all the paragraphs may contain a keyword.

In the dataset, each pair $\left(\mathcal{X}^{(i)}, \mathcal{Y}^{(i)}\right)$ is assigned with the heading title of the chapter or subchapter where it occurs. Therefore, there is a triplet $\mathcal{U}^{(i)} = \left(\mathcal{X}^{(i)}, \mathcal{Y}^{(i)}, \mathcal{Z}^{(i)}\right)$ for each row in the corpus. In the same way, headings are defined as a sequence of $L_{\mathcal{Z}}$ words:

$$\mathcal{Z}^{(i)} = z_1^{(i)}, z_2^{(i)}, ..., z_{L_{\mathcal{Z}}}^{(i)} \tag{6.4}$$

UPDF
WWW.UPDF.COM

6.3 Model Definition

6.3.1 BERT Encoder

Before feeding the paragraphs into BERT, each heading is prepended to the corresponding passage to form the following composite query:

$$\mathcal{Q}^{(i)} = [\,[CLS]\ \mathcal{Z}^{(i)}\ [SEP]\ \mathcal{X}^{(i)}]\,] \tag{6.5}$$

where [CLS] is the *classification token* and [SEP] indicates a *separation token* interposed between the heading and the passage to inform the model of the two different elements in the input sequence. Given that chapter titles and subtitles are often used to summarize the focus of the section in a few words, I believe the model benefits from this additional information, actively looking at the heading to better recognize the importance of candidate keywords for the encoded passage. Thus, each paragraph was paired with the heading of the chapter or sub-chapter where it occurred.

Once the query pass through the Embedding Layer, natural language is first tokenized and then univocally converted into a sequence of N embedding $\boldsymbol{e} = \{e_1, e_2, ..., e_N\}$, so that $f(\boldsymbol{tk}_j) = \boldsymbol{e}_j$, where $\boldsymbol{e} \in \mathbb{R}^{N\times d_{BERT}}$ and $\boldsymbol{tk}_j$ correspondents to the j-th token from the query $\mathcal{Q}^{(i)}$.

It is relevant to notice that BERT has a limited input capacity so that $N \in [1, 512]$. This bounded span contains both the heading and the textbook paragraph. In rare cases, the combination of the two exceeded this limit. When it happened, the paragraph were truncated to respect the length limit.

Concluding this step, $\boldsymbol{e}$ is fed into the BERT encoder that acts as follows:

$$\boldsymbol{g}^{(l)} = LayerNorm(\boldsymbol{h}^{(l-1)} + MultiHeadAtt(\boldsymbol{h}^{(l-1)}, \boldsymbol{h}^{(l-1)}, \boldsymbol{h}^{(l-1)}) \tag{6.6}$$

$$\boldsymbol{h}^{(l)} = LayerNorm(\boldsymbol{g}^{(l)} + FFNet(\boldsymbol{g}^{(l-1)})) \tag{6.7}$$

where $\boldsymbol{h}^{(l)} = \{h_1^{(l)}, ..., h_m^{(l)}\}$ denotes the output of the l-th layer and $\boldsymbol{h}^{(0)} = \boldsymbol{e}$, with $\boldsymbol{h}^{(l)} \in \mathbb{R}^{N\times d_{BERT}}$. The three inputs of the multi-head self-attention layer are query matrix, key matrix and value matrix from left to right. The vector generated by the last layer of BERT will be referred to as $\boldsymbol{h}$ from now on.

In (6.6) and (6.7) the embeddings go through two sub-layers. The first is a multi-head self-attention mechanism, responsible for the attention vector that represents how much each word in the sequence has to pay attention at the other ones, and the second is a standard fully connected feed-forward network. Both combined with a layer normalization and a residual connection.

6.3.2 Bi-LSTM and Linear Classifier

The contextualized word vectors generated from BERT continue into BiLSTM layer and a Linear Classifier stacked on top of it. As said in §4.1, a BiLSTM is a composite model consisting of two LSTM modules: one taking the input in a forward direction, and the other in a backward direction. Formally for any time step t, the output generated by the last layer of BERT h_t follows the following steps:

$$\begin{aligned} \boldsymbol{m}_t^{(f)} &= \sigma(\boldsymbol{h}_t\boldsymbol{W}_1^{(f)} + \boldsymbol{h}_{t-1}\boldsymbol{W}_2^{(f)} + \boldsymbol{b}^{(f)}) \\ \boldsymbol{m}_t^{(b)} &= \sigma(\boldsymbol{h}_t\boldsymbol{W}_1^{(b)} + \boldsymbol{h}_{t-1}\boldsymbol{W}_2^{(b)} + \boldsymbol{b}^{(b)}) \end{aligned} \tag{6.8}$$

UPDF
WWW.UPDF.COM

Here, σ is the sigmoid function and $\boldsymbol{W}_1^{(f)}$, $\boldsymbol{W}_2^{(f)}$, $\boldsymbol{b}^{(f)}$, $\boldsymbol{W}_1^{(b)}$, $\boldsymbol{W}_2^{(b)}$, and $\boldsymbol{b}^{(b)}$ all the parameters of the forward and backward LSTMs defined in §4.1.

Next, the forward and backward hidden states $\boldsymbol{m}_t^{(f)} \in \mathbb{R}^{1\times d_{BiLSTM}}$ and $\boldsymbol{m}_t^{(b)} \in \mathbb{R}^{1\times d_{BiLSTM}}$ are concatenated to obtain the hidden state $\boldsymbol{m}_t \in \mathbb{R}^{1\times 2d_{BiLSTM}}$. The sequence generated by the last layer of the BiLSTM is here denoted $\boldsymbol{m} = \{m_1, m_2, ..., m_N\}$ such that $\boldsymbol{m} \in \mathbb{R}^{N\times 2d_{BiLSTM}}$.

Lastly, $\boldsymbol{m}$ is fed into the Linear Classifier that computes a score $\boldsymbol{y} \in \mathbb{R}$ reflecting the semantic closeness of each token $\boldsymbol{tk}_j$ in the initial sequence respect to the target domain. This is done through a thresholded function:

$$\boldsymbol{y} = f(\boldsymbol{W}_{lc} * \boldsymbol{m}) = f\left(\sum_j \boldsymbol{w}_j * \boldsymbol{m}_j\right)$$

$$\text{where} \quad f(\cdot) = \begin{cases} 1, & \text{if } \boldsymbol{W}_{lc} * \boldsymbol{m}_j > \boldsymbol{k} \\ 0, & \text{otherwise} \end{cases}$$

$\boldsymbol{W}_{lc}$ and $\boldsymbol{k}$ being the parameters of the classifier and the threshold of acceptance to positively classify a token, respectively. For the final version of the model, I set $\boldsymbol{k} = 3$.

6.4 Prediction Post-processing

This section describes the final layer of the pipeline. Once the classification is concluded, the model returns a collection of candidate keywords that, according to the scores calculated after the linear classifier, have high relevance for the passage where they occur. Before comparing against the ground truth, the candidates are further examined using some *principles* that are formulated to align the extracted keywords to the entries produced by professional indexers.

Modifiers Principle: valid terms should not be modifiers

the three principles a included but with short explanation

Modifiers are words or groups of words that describe other words or groups of words. Two common modifiers are adverbs, which describe adjectives, verbs, or other adverbs, and adjectives, which describe nouns or pronouns. This principle comes directly from the assumption that modifiers should not be found as independent index entries (Mulvany, 2013) in order to avoid redundancies in the index. For instance, the adjective "bivariate" means "involving or depending on two variables". Even though the word carries statistical information, it can be associated with many different terms, including "bivariate correlation", "bivariate regression", or "bivariate normal density"; each of them representing a distinct stand-alone concept. A reader looking for one of these entries would need more than one attempt if the word bivariate was repeated many times across the index. Given that the objective is the extraction of keywords that helps defining a precise topic, modifiers are discarded.

Structural Principle: valid terms should reflect the syntactical structures used by professional indexers guidelines.

UPDF
WWW.UPDF.COM

Other than choosing index entries that unambiguously refer to domain-specific terms, indexers are also asked to respect guidelines on the phrasing of index terms. In *Indexing Books* (Mulvany, 2013), the author suggests that main headings and subheadings should generally be nouns, nouns preceded by adjectives or gerunds. To uncover the presence of such patterns in the phrasings favored by professional indexers, I conducted a preliminary analysis of the index terms from the GC.

As shown in 6.2A, more than 90% of the headings created by human indexers count 3 words at most. Given that they rarely exceed this number, the analysis focused on 1-grams and 2-grams. This is also justifiable by the fact that the complexity of possible patterns increases exponentially, i.e. k^n where k is the number of syntactical classes and n the words in the n-gram. Hence, with longer n-grams, it is more challenging to elaborate general rules that apply to each case. In FIGURE 6.2B and 6.2C, it is shown how 1-grams and 2-grams belong to a few syntactical categories. Proven that professional indexers do follow shared compositional patterns, the predictions of the model are filtered based on hard-crafted rules that reflects these syntactical structures.

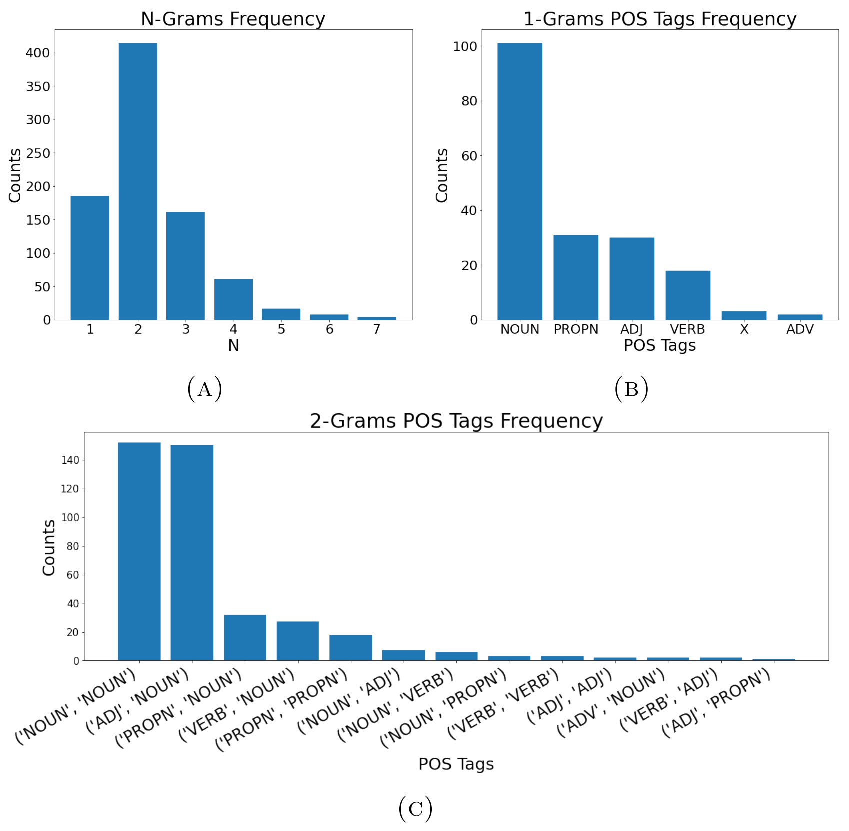


FIGURE 6.2: Three graphs showing recurrent syntactical patterns among index entries created by professional indexers. (A) illustrates how most of the index terms are rarely longer than three words. In (B), POS tags from monograms are compared. As expected only the four classes of nouns, adjectives, verbs, and proper nouns are predominant. Lastly, (C) shows syntactical patterns of bigrams. Although here the variability is greater, there are only two categories that constitute the majority of the distribution.

For 1-grams I limited the prediction to only three major classes, i.e. nouns (NOUN), proper nouns (PROP), and verbs (VERB). Although adjectives (ADJ) represent a big portion of the total distribution (FIGURE 6.2B), they are excluded

UPDF
WWW.UPDF.COM

due to contradiction with the Modifiers Principle. The category of 2-grams presents a distribution with only two relevant classes. Filtering minor classes would have been less effective in this case, hence, to avoid the risk of losing relevant information, all the 2-grams were considered to be valid.

With the length of the n-grams, the number of classes was increasing, making it virtually impossible to find specific filters. Consequently, more general rules were applied to all the other predictions, independently from the number of words in the candidate keywords. Taking into account dependency tags, the first or the last word in an extracted phrases must not belong to the following syntactical classes: coordinating conjunction, punctuation, determiner, preposition or subordinating conjunction, and adposition.

> ***Completeness Principle***: valid terms should not be truncated or incomplete.

Another factor to consider is the tokenization methods used at the bottom of BERT. WordPiece, already described in §4.3.1, split certain words in smaller pieces to optimize the known lexicon in the vocabulary.

During inference, some of these word pieces were recognized to be relevant while the rest of the tokens that compose the term were left out. For instance, in the case of "Bayes" and "##ian", it might happen that only "##ian" is retrieved. This behavior is traceable to the contextual nature of the embedding produced by BERT. Even though "##ian" does not represent a statistical term in any way possible, the first part of the term is relevant to the domain; hence, the second part could benefit from this nearness and therefore be positively classified. Although improbable, there were some cases where only the second half of a relevant term was recognized. To account for it, any character that starts with a double-pound sign, i.e. "##", was removed from the pool of candidate keywords.

UPDF
WWW.UPDF.COM

Chapter 7

Evaluation and Results

In the set of contributions stated in §1, the first one was the implementation of an end-to-end pipeline to construct domain-specific corpus. In order to verify the quality of the dataset, Section 7.1 reports the performance obtained by two deep learning models, i.e. a Bi-LSTM and the model defined in the previous chapter (without post-filtering), in a standard sequence labeling task. As described in §5.1, the corpus is decidedly skewed toward the major class which might undermine its effectiveness. In short, this analysis comes with a twofold aim: (i) corroborate the suitability of the annotation pipeline for training deep learning models, under the assumption that the final performance also reflects the dataset quality; (ii) quantify the benefit of conditioning BERT to detect domain-related words in such an imbalanced scenario.

Even though the identification of positive samples is the first criterion that a good classifier should match, the extraction of keywords is a very distinct problem. Section 7.1 also describes how these two tasks differ and compares the model predictions, now enhanced by the post-filter described in §6.4 with a term bank utilized as ground truth to assess how similar the keywords extracted are to the index entries elaborated by human indexers.

Testing an NLP model from a global point of view is a non-trivial operation, especially when it comes to deep learning algorithms. Corpus-based approaches, like the one described in the previous paragraphs, may not be enough to globally gauge the quality of the work done by the NLP model. While developing the evaluation pipeline, I noticed a pitfall in AKE corpus-based methods: the relative *relevance* of the extracted terms for the document where they occurs is not considered since all the positive samples are valued in the same way. Especially when it comes to information retrieval tasks such as AKE, the notion of relevance has to be taken into account if the goal is understanding the representativeness of the retrieved terms for a passage of interests. For example, the term ”continuous probability distribution” is clearly to be recognized in a statistical textbook. However, its relevance may vary depending on the section of the book where it occurs. Meaning that in the chapter ”The Normal Distribution”, its relevance increases because more representative of the topic being discussed in that passage (in fact a normal distribution belongs to the category of continuous probability distributions). In other words, not only is it necessary to evaluate if a model learns to recognize domain-specific terms but it is also crucial to assess the quality of such terms contextualized to the document where they occur (see in particular (Saracevic, 2007) for a detailed excursus on the

UPDF
WWW.UPDF.COM

concept of relevance in the field of information retrieval).

This thesis addressed this issue by adopting what in research is known as *extrinsic evaluation*. An extrinsic evaluation assesses the model's performance on a downstream application. Given that the goal is to verify the relevance of the retrieved keywords, the predictions generated by the AKE model are tested on a document similarity task, namely the cross-link task (Guerra, Sosnovsky, and Brusilovsky, 2013). Addressing the second contribution given in the first chapter, such predictions are implemented to build VSMs that are used as document representations to identify the most related sections in two textbooks. §7.2 gives a brief description of the experiment and presents the results obtained by the model with two different vectorization techniques. I believe the performance achieved after this evaluation to reflect the ability of the model in capturing relevant information contextualized to the corresponding passages.

If on the one hand, extrinsic evaluation paradigms give immediate feedback regarding the efficiency of the model on a downstream task, on the other hand, they do not permit to determine which aspect of the total architecture is at fault in case of a poor outcome. Conversely, *intrinsic evaluation* is executed as a close-up analysis to measure specific capabilities of the model under examination. Hence, to support the last of the three contributions described in the first chapter, I conducted an intrinsic analysis of the word embeddings generated by BERT to verify the presence of what was previously referred to as lexical knowledge. Section 7.3 aims to contribute to that body of research that tries to explain *how* SOTA LMs work in their internal mechanisms (Rogers, Kovaleva, and Rumshisky, 2020), and it does so by presenting a geometric analysis of the embedding generated by BERT before and after the training phase. The results clarify the effect of fine-tuning BERT on a domain-specific AKE task and uncover if the model also takes into account semantic information during the decision-making phase, giving more insights into the correlation between BERT and DSMs

7.1 Dataset Validation and Keywords Extraction

do not take into account entire KPs and BERT apply a sub-word tokenization which is not used in other DL models, limiting comparison with different models. (increasing the number of tokens result in lower performences in recall)

Classification models are traditionally examined through evaluation metrics calculated from true positives (TPs), false positives (FPs), true negatives (TNs), and false negatives (FNs). Two are True Positive Rate (TPR) and False Positive Rate (FPR), often represented in a Receiver Operating Characteristic (ROC) curve. ROC curves are graphical tools showing TPRs and FPRs at the different thresholds of acceptance: at lower thresholds, more items are classified positively, thus increasing both FPs and TPs. ROC analysis enables to select optimal models and to discard suboptimal ones independently from class distributions. As illustrated in FIGURE 7.1A, two deep models were trained on the generated dataset, and their performance evaluated on a sequence labeling classification task. Sequence labeling is an NLP task that aims to map single words in a sequence to predefined class space. In this case each word is classified as *keyword* or *not keyword*. The baseline is a BiLSTM initialized with word embeddings coming from a pre-trained BERT model.

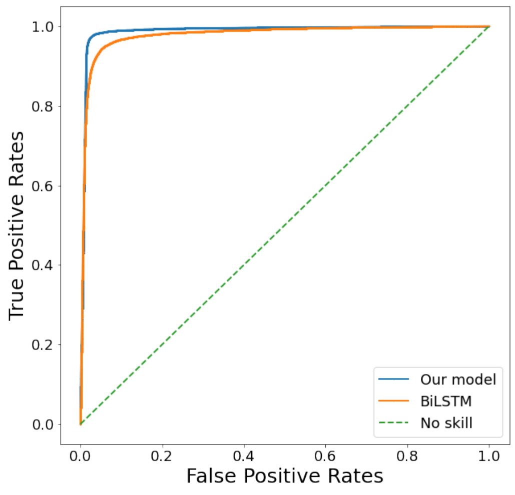
The results of this analysis suggest that the dataset is suitable for the training of deep learning architectures: both models achieved AUC values above 95% (TABLE 7.1), demonstrating that they effectively learned to recognize words from the objective vocabulary distribution. The image also illustrates that the BERT-based model

traditional approach to evaluate AKE models (compare entire keyphrases with gound-truth)

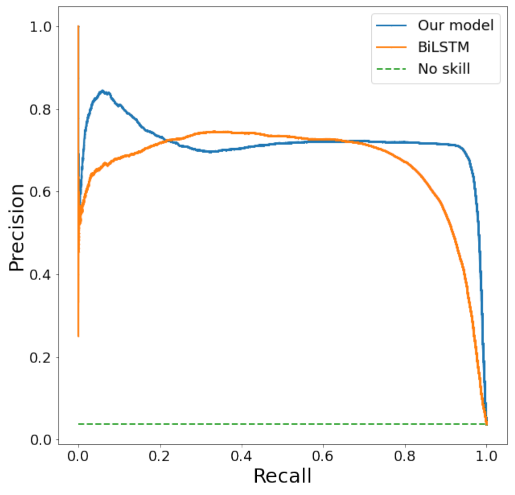
UPDF
WWW.UPDF.COM

slightly outperforms the baseline, giving additional evidence of the benefit gained from conditioning BERT's embeddings on the target domain.

Precision and recall are also popular metrics when it comes to classification tasks. The two can be handly summarized in a precision-recall (PR) curve that, similarly to ROC curves, plots the model performance at different classification thresholds. From the PR curve in FIGURE 7.1, it can be seen that if initially the two architectures achieved comparable performance, after 80% recall the precision of the baseline drops while the BERT-based model still correctly predicts a significant number of relevant instances. The quality of the classifiers is also confirmed by their corresponding AUC values shown in the table below. Differently from ROC curves, PR curves take into account class imbalances. In this case, the "no skill" classifier naively predicts all the samples to be positive.



(A)



(B)

FIGURE 7.1: PR curves and ROC curves of two deep models trained on our generated dataset. (A) shows the quality of the classifiers in terms of FPRs and TPRs. (B) compares them on precision and recall values at different thresholds. The initial drop in precision is due to the levels of threshold: initially only few samples are classified positively due to the high thresholds, so a difference in of one TP or FP could change the curve significantly. Hence, the graph is more volatile in the first step.

The optimal threshold can be identified following different criteria. Speaking of ROC curves, a perfect result would be the point indicating 0% FPs and 100% TPs. However, In real application these performances are never achieved; a simple alternative is to take the point with the maximal sum of true positive and false negative rates. In the case study explored by this research, the PR curve was taken as reference to decide the final threshold as it gives a more straightforward idea of its effect on the precision and recall metrics. According to the standard approach, the optimal threshold value is that that maximizes the F1-score of the classifier. However, desired values of precision and recall may vary on the specific goal of the task. As precision was prioritized, the optimal point was set at 80% recall. TABLE 7.1 summarizes the performance of the two approaches with the results obtained by the models at such a point.

The metrics above described are useful for standard word-by-word sequence labeling tasks, such as POS tagging or NER, but fail to evaluate the model in more

UPDF
WWW.UPDF.COM

%	ROC AUC	P [R=80%]	F1 [R=80%]	PR AUC
Bi-LSTM	97.92	61.66	70.90	66.57
Our Model	98.81	69.85	75.52	71.61

TABLE 7.1: The table presents recall and F1-score metrics at a fixed recall, according to our choice of threshold. AUC values for PR and ROC curves are also included to quantitatively compare the deep models.

complex tasks like AKE: the latter, indeed, requires the classifier to judge when sequences of words are to be positively classified as part of the same term. For instance, the multi-word term "analysis of variance", shortened to ANOVA, represents a specific idea: ANOVA tests are a collection of statistical tools to compare variances across the means of different groups. In this scenario, a naive classifier that independently computes the relevance of each word would rule out "of", splitting "analysis" and "variance" into two terms and overlooking a critical keyword for the goal domain. The correct recognition of multi-word terms is a key aspect to examine before concluding if an AKE algorithm is successful in the task or not.

Motivated by this consideration, I developed a more fitting evaluation approach for AKE tasks that gauges the quality of the predictions not based on the simple ratio of TPs and TNs, but on two pre-built banks of terms, respectively measuring the *Local Recall* and *General Precision* of a classifier.

Local Recall, similarly to the homonym metric, informs about the fraction of index terms correctly retrieved. Local Recall is calculated using a segment of the GC: the ground truth is obtained by restricting the Global Corpus to those index terms included in the textbook/s used in the test set. I did not consider all the GC for a simple reason. Given the non-homogeneity distribution of index terms across textbooks (see FIGURE 5.1A), some index terms could be missing from the indexes of the test textbook/s and therefore be unretrievable.

General Precision, on the other hand, is meant to inform about the correctness of a prediction regardless of the indexes of texbook/s included in the test set. In this case, the ground truth was a term bank that follows the same structure of the GC, but it was expanded with additional index terms. This extended term bank, which I decided to call Global Corpus Plus, shortened GC+, counts on two more books with their index terms and all the subheadings discarded during the filtering phase described in §5.3.1.

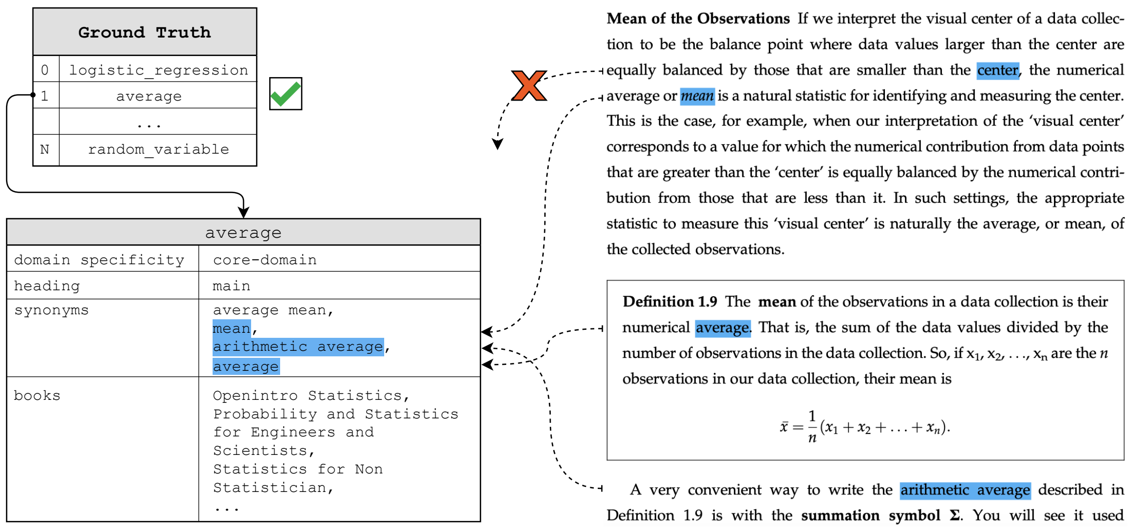
The computation of Local Recall and General Precision goes as follows. After lemmarizing both the ground truth and the extracted terms that passed the post-filtering phase, Local Recall and General Precision are calculated through exact match between the model predictions and the `synonyms` of the index terms in the respective ground truth:

$$\boldsymbol{LocalRecall} = \frac{\sum_{i=0}^{|P|} p_i}{|P|} \qquad \boldsymbol{GeneralPrecision} = \frac{\sum_{i=0}^{|P|} p_i}{|P|}$$

$$\text{where } \; p_i = \begin{cases} 1, & \text{if } p_i \in GC_d \\ 0, & \text{otherwise} \end{cases} \qquad \text{where } \; p_i = \begin{cases} 1, & \text{if } p_i \in GC+ \\ 0, & \text{otherwise} \end{cases}$$

UPDF
WWW.UPDF.COM

with P being the pool of predictions generated by the model and GC_d the subgroup of the Global Corpus that contains only index terms in the document d.



Ground Truth	
0	logistic_regression
1	average
	...
N	random_variable

average	
domain specificity	core-domain
heading	main
synonyms	average mean, mean, arithmetic average, average
books	Openintro Statistics, Probability and Statistics for Engineers and Scientists, Statistics for Non Statistician, ...

Mean of the Observations If we interpret the visual center of a data collection to be the balance point where data values larger than the center are equally balanced by those that are smaller than the center, the numerical average or *mean* is a natural statistic for identifying and measuring the center. This is the case, for example, when our interpretation of the 'visual center' corresponds to a value for which the numerical contribution from data points that are greater than the 'center' is equally balanced by the numerical contribution from those that are less than it. In such settings, the appropriate statistic to measure this 'visual center' is naturally the average, or mean, of the collected observations.

Definition 1.9 The **mean** of the observations in a data collection is their numerical average. That is, the sum of the data values divided by the number of observations in the data collection. So, if $x_1, x_2, \ldots, x_n$ are the n observations in our data collection, their mean is

$$\bar{x} = \frac{1}{n}(x_1 + x_2 + \ldots + x_n).$$

A very convenient way to write the arithmetic average described in Definition 1.9 is with the **summation symbol** $\boldsymbol{\Sigma}$. You will see it used

FIGURE 7.2: The computation of Local Recall and General Precision is based on a term bank. Each terms is considered to be retrieved if at list one of the `synonyms` is identified in the body of the book. This generates one TP. Viceversa, extracted terms that do not appear in any of the `synonyms` lists from the index terms contained in the groud truth generate FPs. In this specific case the model prodeces one TP and one FP

Given that the General Precision is calculated through an exact match with the terms in the `synonyms` list of each index entry, the initial GC used to label the dataset was expanded to increase the coverage of wordings for each index term. The two books were not used for the fine-tuning phase due to system constraints: with more data, I noticed that the time of training increased significantly. However, their index entries are still useful to increase the number of terms in the ground truth and, at the same time, extend the `synonyms` of the already existing index entries. As reported in §5.3.1, many subheadings were not included in the GC during the labeling phase due to the presence of long phrases in the list of `synonyms`. Although there it was crucial to control which parts of the text were labeled, the presence of such phrases was not an issue at this stage. Given that the model was trained to recognize short n-grams, I never noticed long phrases among the predictions. Hence, the risk of incorrect TPs was minimal.

Table 7.2 presents the performance obtained by the BERT-based model after 20 epochs, comparing them with other unsupervised and deep approaches. TF-IDF (Gerard Salton and McGill, 1983) and LDA (Blei, A. Y. Ng, and Jordan, 2003) are popular unsupervised methods that rely on statistical features to assign a value of representativeness to each term. While the former only extracts monograms, LDA can retrieve multi-word keywords. LDA splits documents into topics, each represented by a group of keywords. I extracted a number of topics equal to the section in a book. From each of these, only the top two scored keywords were kept. Among the family of graph-based algorithms, TextRank (Mihalcea and Tarau, 2004) and TopicRank (Bougouin, Boudin, and Daille, 2013) were evaluated. With TextRank and TopicRank I get the 10-highest scored candidates as keywords for each section

UPDF
WWW.UPDF.COM

and topic, respectively. Finally, the experiment involved Key-BERT[1] to extract relevant monograms and bigrams from the sections. KeyBERT relies on a pre-trained version of the Transformer encoder to find keywords through similarity scores computed between word embeddings and a vector representation of the whole passage of interest. Hence, this comparison also served to uncover the benefit of fine-tuning BERT on the target domain. All the baselines except for TFIDF and KeyBERT were implemented using the Python Keyphrase Extraction (PKE) package[2].

%	General Precision m (sd)	Local Recall m (sd)	F1-score m (sd)
TF-IDF	20.59 (7.33)	35.48 (7.48)	26.06 (7.40)
LDA	36.67 (9.76)	31.85 (8.51)	34.09 (9.09)
TextRank	12.99 (3.45)	27.41 (10.94)	17.62 (5.26)
TopicRank	39.54 (8.94)	49.04 (13.51)	43.78 (10.76)
Key-BERT	26.23 (6.08)	52.35 (14.52)	34.95 (8.57)
Our Model	**52.88 (10.97)**	**67.19 (9.36)**	**59.18 (10.10)**

TABLE 7.2: Performance of three families of AKE algorithms calculated with our ad-hoc evaluation metrics.

The results show that the fine-tuned BERT-based model outperformed all the baselines by a large margin. Moreover, it can be seen that Key-BERT performed poorly, demonstrating the benefit of fine-tuning BERT on the proposed dataset.

A comparison of the results in TABLE 7.1 and TABLE 7.2, however, shows a significant drop in performance. With the only exception of our model, all the tested algorithms showed a low precision. Even though our architecture achieved higher results, its precision slightly above 50% reveals a large number of predictions that do not find a corresponding match in the GC+.

7.2 Extrinsic Evaluation

the extrinsic evaluation is kept to support the idea of relevance of extracted KP. However many datils of the implemenattion will be assumed to be known (e.g. TFIDF or BM25)

This experiment was developed to elaborate on the second contribution and assess the relevance of the extracted keywords for the the books in the test set. The problem to solve is the generation of a Vector Space Model (VSM) to represent different sections in two textbook. The idea is that similar sections share more terms, and this results in similar vector representations. The Cross-link task was developed by (Guerra, Sosnovsky, and Brusilovsky, 2013) to create accurate linking of textbooks passages. The level of granularity with which to solve the task is variable: a section could mean a whole book like a single sentence. Following the approach adopted by past research on this task, I divided each document accordingly with its corresponding table of contents. The table of contents is where authors list the chapters of the book, alongside their page numbers. This division, in fact, implicitly suggests how an author decided to split the book into major topics.

[1] https://maartengr.github.io/KeyBERT/
[2] https://boudinfl.github.io/pke/build/html/index.html

UPDF
WWW.UPDF.COM

The evaluation approach compares the mapping created using the generated VMSs against an "ideal" one, entrusted to two expert in the field of Statistics. The annotators could map every section of a book to zero or more parts of the second book. The strength of this connection was indicated with a *level of relevance* and a *level of confidence*, both ranging from 1 to 3. The final score is calculated as the product of these two values.

The evaluation pipeline is structured as follows. First, the full model defined in Chapter 6 is used to extract relevant terms for each of the sections in the two books. Second, the terms belonging to these VSMs are assigned with a score that indicates the relevance of the term for that specific passage. For this step, I opted for two vectorizing algorithms, i.e. TF-IDF and BM25. The former measures local Term Frequency (TF) and global Inverse Document Frequency (IDF) to determine how much relevant a query is.

$$\begin{aligned} TF-IDF(\boldsymbol{d},\boldsymbol{q}) &= TF(\boldsymbol{q},\boldsymbol{d}) * IDF(\boldsymbol{q}) \\ \text{where } \; TF(\boldsymbol{q},\boldsymbol{d}) &= \frac{f(\boldsymbol{q},\boldsymbol{d})}{\sum_{q' \in d} f(\boldsymbol{q}',\boldsymbol{d})}\,, \\ \text{and } \; IDF(\boldsymbol{q}) &= \log\left(\frac{|\boldsymbol{D}|}{|\boldsymbol{d} \in \boldsymbol{D} : \boldsymbol{q} \in \boldsymbol{d}|}\right) \end{aligned} \tag{6.1}$$

where $f(\boldsymbol{q},\boldsymbol{d})$, $|\boldsymbol{D}|$, and $|\boldsymbol{d} \in \boldsymbol{D} : \boldsymbol{q} \in d|$ are the the raw count of a term in a document, the total number of documents in the corpus, and number of documents where the query $\boldsymbol{q}_i$ appears .

BM25 returns numerical values as a linear weighted combination of scores for each word that constitute a term of interest. The algorithm is based on TF and IDF values but improves on the previous approach accounting for document length and term frequency saturation. Given a query $\boldsymbol{q}$ and a document $\boldsymbol{d}$ the total score is calculated as such:

$$\begin{aligned} BM25(\boldsymbol{d},\boldsymbol{q}) &= IDF(\boldsymbol{q}) * \frac{TF(\boldsymbol{q},\boldsymbol{d}) * (\boldsymbol{k}+1)}{TF(\boldsymbol{q},\boldsymbol{d}) + \boldsymbol{k} * (1 - \boldsymbol{b} + \boldsymbol{b}\frac{|\boldsymbol{d}|}{avgdl})} \\ \text{where } \; avgdl &= \sum_i \frac{|\boldsymbol{d}_i|}{|\boldsymbol{D}|}\,, \\ \text{and } \; IDF(\boldsymbol{q}) &= \log\left(\frac{N - DF(\boldsymbol{q}) + 0.5}{DF(\boldsymbol{q}) + 0.5}\right) \end{aligned} \tag{6.2}$$

being $|\boldsymbol{d}|$ and $DF(\boldsymbol{q})$ the length of the document in words and the number of documents that contain the word $\boldsymbol{q}$, respectively. The hyperparameters were set to $\boldsymbol{k}$=1.2 and $\boldsymbol{b}$=0.75. As regards multi-word terms, I decided to treat them as atomic elements, calculating their terms frequency and document frequency scores based on the count of the term as a whole.

Using these scores I obtain a list of terms sorted on their relevance for each section. These ordered lists allow us to evaluate the model through a ranking metric,

UPDF
WWW.UPDF.COM

namely the normalized Discounted Cumulative Gain (nDCG):

$$nDCG = \frac{DCG}{iDCG}$$
$$\text{where } \; DCG = \sum_{i=1}^{p} \frac{\boldsymbol{s}_i}{\log_2(i+1)} \; ,$$
$$\text{and } \; iDCG = \sum_{i=1}^{|S_p|} \frac{\boldsymbol{s}_i}{\log_2(i+1)}$$

$\boldsymbol{p}$, $|\boldsymbol{S}_p|$, and $\boldsymbol{s}_i$ representing the ranked position of an item i according to the similarity metric, the ranked position of an item i according to the relevance scores, and the score of an item i, respectively.

TABLE 7.3 shows the performance of various models tested on the Cross-link task. From each family of AKE algorithms examined in the previous experiment, I left out those models that performed worse and, for the remaining algorithms, computed average nDCG scores together with their variance. The terms extracted by the baselines receive a similar post-filtering phase to §6.4 as only sequences of nouns or adjectives were accepted.

%	**nDCG**			
	TFIDF		**BM25**	
	m	**sd**	**m**	**sd**
LDA	58.7	28.7	47.6	**28.5**
TopicRank	68.3	30.4	64.9	30.7
Our Model	**77.0**	30.5	70.8	30.3

TABLE 7.3: Comparison across families of algorithms of the performance calculated in the Cross-link task. Both mean and variance were included to give a clearer idea of the effectiveness and robustness of the various models.

The standard deviations of all the models' performance showed no significant difference, with the only exception of the LDA model paired with BM25 which obtained the lowest variance by 2 percentage points Conversely, the BERT-based architecture achieved an average nDCG greater than the other algorithms, with a positive margin of 8.7% from the second-best method. The score is also greater than those reported by (Alpizar-Chacon and Sosnovsky, 2020) who used index terms directly extracted by human annotators.

Another interesting finding was that, using TFIDF to vectorize the extracted terms proved to be a better approach compared to BM25 in all the tested models. Although the difference in the variance of the results obtained with the two algorithms is minimal, it can be noticed that the average score differs consistently across the three tested approaches: using TFIDF always resulted in better performance. The interpretation of these results can be found in Chapter 8.

UPDF
WWW.UPDF.COM

7.3 Intrinsic Evaluation

One general shortcoming of standard ways to evaluate deep models is that they are tested on benchmark datasets without being supported by exhaustive explanations that elaborate on the success or failure of the task. Recognizing the value of intrinsic evaluation in explaining deep architectures, here I elaborate on the third contribution, performing an analysis of the vector space generated by BERT before and after fine-tuning it on a domain-specific corpus. A study of BERT's embedding offers the opportunity to uncover patterns that explain its behavior, and thus establish guarantees that the performance will continue to be consistent when deployed in future applications. In particular, this section elaborates on the capability of BERT in capturing semantic similarities in a domain-specific scenario.

In §3.4, I provided background knowledge on DSMs, including BERT to this category of models. One of the key properties of DSMs is that the information enclosed in the vector embeddings is *graded* (Boleda, 2020), meaning that it is expressed in continuous semantic space, in which semantic relations are modeled as geometric relations. Through this property, DSMs enable to compute a measure of similarity between words that span a continuous range of values. Past research proved that pre-trained LMs like BERT capture a rich pool of features from text data: linguistic features (e.g. POS tag, dependency tags, etc); syntactic features (subject-verb agreement, coreference, etc); and morphological features (suffix and prefix). To bring an additional contribution to this branch of research, I argued that BERT is also able to capture lexical knowledge if fine-tuned on a specific goal domain. More specifically, I believe that training BERT for relevant term extraction centers its lexical knowledge around a specific vocabulary.Thus, the location of the resulting embeddings in the vector space reflects this acquired expertise as more similar terms are placed in closer regions.

To give proof of this, I compared the embedding generated by BERT before and after fine-tuning. Since the encoder model was pre-trained and fine-tuned with natural language, feeding the model with uncontextualized terms would have produce nonoptimal representations (Chronis and Erk, 2020). For this reason, aggregate vector representations were created by averaging BERT's embeddings for the single term t in different contexts c_i. More formally:

$$\boldsymbol{t} = mean(t_{c_i}, ..., t_{c_n})$$

where $\boldsymbol{c}_i$ is the sentence where the target occurs, and $\boldsymbol{t}_{c_i}$ is the token embedding for t in the context c.

Following (Lenci et al., 2022), I set a minimum sentence length of 4 tokens, and a maximum bound at 21 tokens. For each target $\boldsymbol{t}$, up to 100 sentences were selected to produce aggregated representations (Vulić et al., 2020). For some terms, there was a number of sentences inferior to this threshold.

In FIGURE 7.3, the heatmaps show cosine similarity values between word vectors belonging to the "core-domain" and "in-domain" categories. The difference between 7.3a and 7.3b clearly illustrates that terms related to the goal domain present a greater similarity after the training phase. A non-parametric Mann-Whitney U test (McKnight and Najab, 2010) was also run to validate these results, finding a strongly significant correlation (TABLE 7.4). Since the geometric distance between word vectors of domain-specific terms decreases, I decided to call this phenomenon

WWW.UPDF.COM

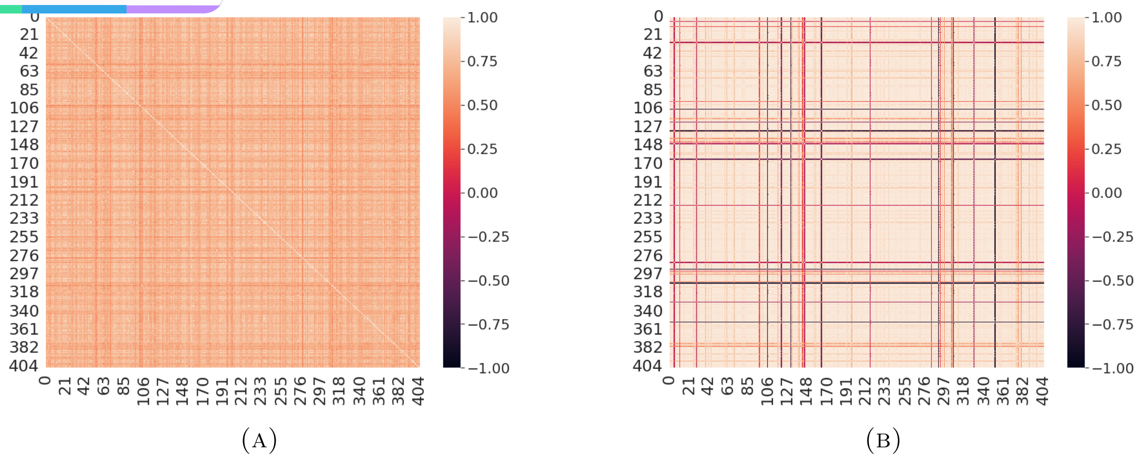


FIGURE 7.3: Heatmaps displaying cosine similarities between BERT's word embeddings of 405 *core-domain* and *in-domain* terms (A) before and (B) after fine-tuning on a domain-specific dataset.

pull-in effect. Interestingly, in FIGURE 7.3a can be observed some outliers: while the general trend seemed to follow the above-described effect, some terms were instead pushed away from the others. In particular, 21 terms have a lower average similarity after the training, and 11 of these showed mean cosine similarity values below zero (see TABLE B.1 in the appendix for the full list). I believe that for part of them this can be attributed to the lower number of sentences used to generate aggregated representations. With less context, it is plausible that the model does not fully capture the meaning of the specific term. However, one term, i.e. "p-value", showed a negative score after the training even though its vector representation received enough context.

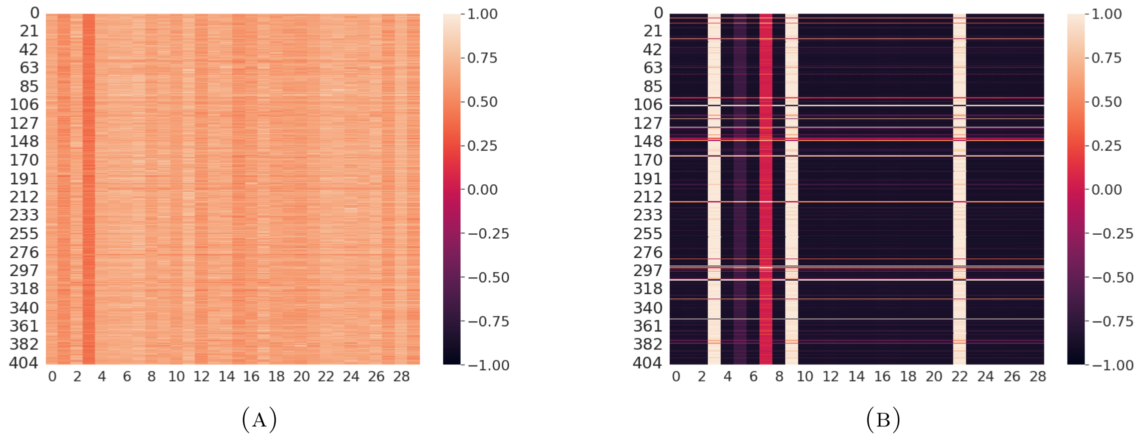


FIGURE 7.4: Heatmaps displaying cosine similarities between BERT's word embeddings of 405 domain related terms and 29 *out-of-domain* terms (A) before and (B) after fine-tuning on a domain-specific dataset.

Other than recognizing relevant terms, a good quality model should also successfully identify out-of-domain terms. Therefore, the previous analysis was repeated on word embeddings but this time between domain-relevant terms and "out-of-domain" terms. From the nine textbooks included in the dataset, 29 out-of-domain terms were available to compare against relevant ones. As illustrated in FIGURE 7.4, the

UPDF
WWW.UPDF.COM

difference between vectors of these two categories increases even more than the previous setting. Accordingly, I called this phenomenon *push-out effect*. Also in this case few terms were not following the general trend. Four terms, in particular, showed a positive average similarity with domain-related terms after fine-tuning BERT. I believe that these terms were placed closer to the domain of Statistics because words like "process" or "noise" are ambiguous if taken out of context. Hence, they can be found in many different domain, and at the same retain a certain degree of relevance. (all these terms are listed in TABLE B.2)

Effect	**Cos. Sim. Before**	**Cos. Sim. After**	**t**	**Sig.**
pull-in	0.72	0.85	9945	≪.001
push-out	0.63	-0.52	161949	≪.001

TABLE 7.4: Results of two one-tailed U-test within domain-related terms and between domain-related and out-of-domain terms.

UPDF
WWW.UPDF.COM

Chapter 8

Discussion

In this thesis, I have presented an end-to-end pipeline for the creation of domain-specific AKE corpora together with a thorough evaluation of a BERT-based model trained on the generated dataset. The three research contributions aimed to: (i) explore the feasibility of a novel AKE dataset construction framework as an alternative to traditional annotation in order to reduce the time and costs of production in domain-oriented scenarios; (ii) probe the effectiveness of BERT in recognizing meaningful terms in long-length documents for the generation of VSMs applied to document similarity tasks; (iii) conduct a spacial analysis of the vectorial space populated by BERT's embeddings before and after training to better understand the effect of fine-tuning on the model output and the possible connection between BERT and the field of distributional semantics.

The first experiment tested two deep models on a standard sequence labeling task where each token was to be binary classified in *keyword* or *non-keyword*. A ROC and PR curve showed the achieved performance, with the BERT-based model returning higher AUC values in both cases. Additionally, it proved to be more robust than the BiLSTM fed with pre-trained embeddings: while the precision of the latter plummed at the end of the second graph, the former maintained constant levels of precision even at greater thresholds of acceptance. I believe these results prove two points. First, the dataset is able to convey the target vocabulary distribution giving encouraging signals to be suitable for AKE deep algorithms. Second, since the model showed better performance in both graphs, it can be deduced that conditioning BERT's word embeddings to the target domain brings a significant benefit to solving the task. Subsequently, the terms extracted by both unsupervised and supervised approaches were compared with two phrase banks of statistical terms. Even though the fine-tuned model outperformed all the other baselines, the performance were considerably lower than the previous examination, with precision being slightly above chance. The rest of the algorithms performed suboptimally with values of recall and precision that never reached 50%. I suspect two factors — other than the errors that the models naturally make — to be responsible for this trend. Firstly, using an exact match paradigm while evaluating prediction ensures to have reliable results but, in determining cases, it is also prone to incorrect FPs since it does not consider semantic similarities between two terms. For instance, if the term "mean squared error", retrieved by an AKE algorithm, and "mean squared deviation", included in the GC+, are compared, an exact match approach results in a FP outcome even though the two terms represent the same concept. Secondly, the

UPDF
WWW.UPDF.COM

fact that the second half of the distribution pictured in FIGURE 5.1B contains terms that appear less than 50 times in the whole corpus makes it challenging to inform the model about their importance in the context. Consequently, as the proportion of these phases decreases, the models see fewer instances and rarely assign high representativeness scores.

Section 7.2 elaborated on the model performance from an extrinsic point of view: the best models from the previous investigation were tested on the Cross-link task proposed by (Guerra, Sosnovsky, and Brusilovsky, 2013) to verify their effectiveness in retrieving representative keywords from each section of two academic textbooks. The BERT-based architecture showed the best results, improving on both baselines and past approaches (Alpizar-Chacon and Sosnovsky, 2020) by a significant margin. In particular, the authors of the research obtained a unified pool of terms of 1611. Differently, the model proposed in this thesis returned a smaller set of 1179 keywords extracted from the body of the examined textbooks. I think the reduced throughput of predictions contributes to the better performance of the latter approach. It is plausible that while being trained on the 8 books, the parameters update of the model was mainly affected by the distribution of those index terms that more often appear in the trained set. For the same reason given at the end of the previous paragraph, it left out "rare" terms that might have decreased the similarity of the vector representations between two related sections. In other words, it retrieved those terms that more commonly represented textbook topics, ignoring specific terminology or peripherical information. This would also explain the lower average variance.

Surprisingly, TFIDF proved to be a better algorithm to vectorize the extracted terms: with BM25, all the models performed worse. This is plausibly caused by the effect that section length has on the scoring algorithm: (Lv and Zhai, 2011) reported that in the case of very long documents, i.e. 2000 words upwards, BM25 stops being effective as it tends to over-penalize wordy passages. The reason for this stays in (6.2). The document length $|d|$ is inversely proportional to the score assigned by BM25. This means that for high $|d|$, the score will approach zero as if the query q was not occurring in the document. TFIDF, being independent of this factor, does not penalize long documents like BM25. In the explored case study, sections are chapters or subchapters of a textbook that can easily exceed 2000 words. More specifically, on average each section was 1239 and 3514 words long respectively for the fist and second books used in the Cross-link task.

More interesting is the fact that the results from the last two investigations are somewhat counterintuitive. While half of the predictions are classified to be wrong, using those same predictions from a document similarity task produced the best performance. A possible interpretation is that many terms classified as false positives are, on the contrary, domain-related terms. The reason behind this misclassification stays in a lack of ground-truth terms in the phrase bank. Motivated by this discrepancy and interested in understanding the real quality of the predictions, I conducted a qualitative analysis of the extracted terms following those three principles defined in §6.4. In accordance with the hypothesis formulated just before, part of the predictions were indeed misclassified as false positives since their counterparts were not included in the GC+. The presence of "novel" terms reveals an interesting property of these statistical models that seem not to be bounded by what they were shown during training. Due to the deep nature of BERT-like architectures, it is difficult to

UPDF
WWW.UPDF.COM

give a single explanation for this behavior. However, I think that the model learns what I previously called lexical knowledge of the target domain. In other words, it does not only memorize the examples seen in the dataset, but it is also able to understand the semantic domain of terms based on lexical patterns and contextual information, and therefore retrieving domain-related terms regardless if the wording was unseen or not. Although the use of BERT to generate novel terms was not part of the research contributions, I fed the model with two sentences containing the novel terms "Wishart distribution" and "Poisson point process". Interestingly, it correctly retrieve them; probably due to the seen words "distribution" and "Poisson" which indicated the model that those terms belong to the statistical domain. The same operation was repeated with more terms from the ISI Glossary that were not present in the GC. The corresponding passages were extracted from Wikipedia abstracts. The table C.1 in Appendix C summarizes the short experiment. Although many unseen terms were recognized, this behavior was inconsistent with less than half of the terms recognized. This is not surprising since the training of the model prioritized precision, and therefore not specialized in discovering such novel terms.

Finally, I studied the word space generated by the fine-tuned BERT to investigate the presence of geometric properties characteristic of DSMs and determine if the classification of the terms was also following semantic principles. Distributional semantic models (DSM) dynamically build semantic representations — in the form of high-dimensional vector spaces — and through these create a word space in which similar terms are grouped more closely. The statistical test used to analyze cosine similarities of embeddings before and after the training phase showed strong significant evidence that domain-specific terms are pulled closer to each other while out-of-domain terms are pushed away from the latter, reflecting spacial properties of DSMs. To the best of our knowledge, no research addressed the effect fine-tuning has on the geometric properties of the resulting word space. (Mickus et al., 2019) examined a pre-trained BERT to better understand to what extent similar words lie in similar regions of its semantic space. The paper concludes with a potential connection between BERT and DSMs but, before giving a certain answer, the authors emphasize the need for further research on some caveats which might confute this theory. (Lenci et al., 2022), on the other hand, classified BERT as DSM based on the fact that the produced embeddings encode meaning as a function of distributional properties. A comparison of the vector space between BERT and traditional DSMs revealed semantic spaces that are rather different. Moreover, the Transformer encoder largely underperformed in the intrinsic tasks in which the DSMs were tested.

Although these studies pave the way for a joint interpretation of Transformer encoders as DSMs, they did not analyze BERT's behavior on a knowledge-driven task such as AKE and the role that fine-tuning has in it. The conclusion derived from the results obtained in §7.3 is that the fine-tuned model can be interpreted as a *region model* (Lenci, 2018). Region models are DSMs that tend to identify semantic neighbors in terms that are topically similar, i.e. terms belonging to the same semantic domain, like "Bernoulli distribution" and "analysis of variance".

8.1 Implications

The findings above-described have both practical and theoretical implications. From a pragmatic point of view, the annotation framework comes with a twofold advan-

UPDF
WWW.UPDF.COM

tage. Firstly, the labeling process requires minimal human intervention, overcoming the already mentioned limitations of standard approaches. Secondly, although this thesis operated in the field of Statistics, the pipeline can be extended to other academic domains, e.g. Biology, Physics, Medicine, Law, etc. The only requirement is a sufficient number of textbooks provided of an index. If this is met, the result is a dataset that can be used to train supervised learning models at close to no cost of production. Moreover, the trained model is not bounded to academic textbooks but can be applied to any string of text from which domain-relevant terms are to be extracted. In short, this pipeline lightens the burden of dataset construction for AKE algorithms, while at the same time maintaining all the benefits and the effectiveness of supervised learning approaches: if the same performance is found in other domains, it would be possible to train multiple "ad-hoc" models focusing on a specific academic field and at the same time cover a broader range of topics without additional effort.

One unanticipated finding was the prediction of keywords that did not appear in the GC used to label the training set. These novel terms suggest that BERT can be used to explore the target domain generating original key terms not included in the groud-truth. These novel terms could then be used to complete the groud-truth itself, adding new terms or expanding the list of synonyms of existing terms. More formally, this process is known as Knowledge Base Completion and involves the identification of missing entities in the Knowledge Base by reasoning about the information present in the knowledge base.

Lastly, I gave proof of a theoretical property of a domain-specific BERT: its embeddings are organized according to their similarity with the target domain. Proving BERT as DSM has different advantages. First, it promotes the idea that a fine-tuned BERT can learn semantics beyond pure statistical patterns. Second, based on the fact that the capability of recognizing semantic domains aid the extraction of keywords, it is reasonable to think that the developed approach could be applied to other domains with performance similar to the one reported in the explore case study.

8.2 Limitations and Future Works

This study was limited by the following points. The annotation of the dataset is based on a PDF parser which occasionally does not recognize non-textual elements (e.g. images, tables, diagrams, etc.) generating noise strings that are difficult to identify and remove. The heuristics described in §5.3.2 rule out the majority of them but the system needs to be perfected to completely eliminate the problem.

In cleaning the dataset, it was necessary to manually verify that all the index entries used to label the documents were related to the goal domain. The process did not require any particular effort thanks to the assumption made in Section 5.3.1, but it indeed represents a bottleneck that limits the capacity of the whole framework. Future research will be dedicated to working around this manual step to make the whole pipeline completely automatic. Moreover, although the annotation framework is designed to be applicable in other domains, this option was not explored to better focus on the analysis of a single one and study the response that deep models have on it. Given the fact that the vocabulary distribution between academic fields varies significantly, it would be interesting to verify the effectiveness of the AKE pipeline

UPDF
WWW.UPDF.COM

in different scenarios and study how the outcome differs.

The evaluation of the extracted terms is a problematic step that still needs to be improved. Relying on an exact match ensures robust identification of TPs but the approach is also prone to many incorrect FPs that lead to underestimating the model's capabilities. Alternative approaches could include semantic similarity metrics, such as cosine similarity or euclidean distance, to verify a possible match in the ground truth. This disambiguation is crucial to calculate the real precision of an algorithm.

Due to system memory constraints, it was not possible to train the model on more data. To compensate for this, the evaluation was conducted on cross-validation. However, the difference between the train and test sets was at times notable. As explained before, even if all the textbooks were elaborating on Statistics, the specific domain focus could vary. Two examples are "Probability and Statistics for Computer Science" and "Statistics for Scientists and Engineers" which deals with statistics in two different fields of study. Expanding the training set would increase the similarity between the two, and the overall performance of the model would benefit from it.

Lastly, the analysis of the vectorial space did not account for specific geometric properties within the target region but rather on proving that fine-tuning brings topically similar terms closer to each other. The graphs in Appendix C (FIGURE D.1 and FIGURE D.2) shows embeddings ante and post training. It seems that that some terms arrange in more defined clusters. Future directions could explore the relationships that stand within terms of the same domain and explore methods to verify the existence of smaller clusters representing individual topics in the source text. Moreover, a following research could experiment with multiple domains or combinations of these ones, and understand if it would be possible to shape the semantic space in something similar to what is shown in FIGURE 3.2.

WWW.UPDF.COM

Chapter 9

Conclusion

This project was undertaken to design a ”cheaper” framework for the annotation of domain-oriented AKE corpora that could be applied to divers academic domains. Hence, a BERT-based model was tested to extract meaningful terms from textbooks, and the retrieved keywords were used for the creation of document representations in the form of VSMs. Moreover, this thesis proposed an analysis of the word space generated by BERT before and after the fine-tuning phase. Overall, the keyword extraction pipeline showed encouraging results that uncovered the effectiveness of the generated dataset and the advantage of conditioning BERT on it. The most salient conclusions of the research are here summarized in the following points.

Textbooks as knowledge bases for datasets construction. The initial idea that started this research was to make use of academic textbooks as a knowledge base to annotate domain-specific corpora through the entries included in the corresponding indexes. Diverse tests were conducted to verify if the approach is valid. A sequence labeling task showed good results with high AUC values, particularly the BERT-based model achieved a 98.81%. Although this should prove that the dataset gives a representative picture of the target lexical distribution, comparing the extracted keywords and keyphrases with the ground truth, resulted in lower performance: after computing an optimal threshold of acceptance, the model achieved precision and recall of 69.85% and 80.00% in the sequence labeling task while in the second experiment the performance dropped to 52.88% and 67.19%, respectively. Rather than an inefficacy of the model to retrieve meaningful terms, I think these results indicate the need for a better evaluation paradigm. Indeed, there were some domain-related terms predicted by the model which were not part of the ground truth. In short, I believe that the labeling of the textbook is properly executed, opening up ways to create datasets in many different domains at reduced costs, but future efforts have to be directed towards automatize such process and engineering a more reliable evaluation system to fairly analyze the AKE algorithms tested on generated corpus.

BERT for VSMs. The keywords extracted by the tested models were then converted to numerical vectors and used to create VSMs of the chapters in two textbooks. Hence, cosine similarities were calculated for all the pairs and the algorithms were evaluated using nDCG to find the most similar sections in the two books. The ranked lists were compared with the links that two experts created after solving the same task. Interestingly, the fine-tuned model achieved the best performance, i.e. 77.0%, with a notable margin with respect to the baselines. This somewhat contradicts what was found in the previous examination, letting us believe that the quality of the extracted terms is higher than what the precision in the second experiment would make us think. The achieved performance confirms the benefit of the training BERT on the generated dataset over unsupervised methods and pre-trained versions of the encoder, supporting the effectiveness of deep models in the generation of VSMs.

Is BERT a DSM? Yes, if fine-tuned on a specific domain assumes behaviour similar to a regional DSM which organizes the entities populating its word space according to topical properties. Two interesting phenomena emerged after the fine-tuning. First, domain-related terms are pulled closer to each other. Second, out-of-domain terms are pushed away from the major cluster (see FIGURE D.3 and FIGURE D.4 for a visual representation). Based on the idea that the semantical principles that BERT can be generalized to diverse domains, I believe that the implemented system could be applied successfully to other academic domains. The discussion section presents some hypotheses on how these new properties could be used for topic modeling applications.

The last part of the thesis is dedicated to highlighting the limitations that could not be addressed for time or resource constraints. Together with these, future research is directed towards alternative evaluation approaches and a more in-depth analysis of word spaces generated by BERT.

UPDF
WWW.UPDF.COM

Bibliography

Alami Merrouni, Zakariae, Bouchra Frikh, and Brahim Ouhbi (2020). "Automatic keyphrase extraction: a survey and trends". In: *Journal of Intelligent Information Systems* 54.2, pp. 391–424.

Alammar, Jay (2018). "The Illustrated Transformer." In: URL: http://jalammar.github.io/illustrated-transformer/.

— (2019). "The Illustrated Transformer." In: URL: http://jalammar.github.io/illustrated-bert/.

Alpizar-Chacon, Isaac and Sergey Sosnovsky (2020). "Order out of chaos: Construction of knowledge models from pdf textbooks". In: *Proceedings of the ACM Symposium on Document Engineering 2020*, pp. 1–10.

— (2022). "What's in an Index: Extracting Domain-Specific Knowledge Graphs from Textbooks". In: *Proceedings of the ACM Web Conference 2022*. WWW '22. Virtual Event, Lyon, France: Association for Computing Machinery, pp. 966–976. ISBN: 9781450390965. DOI: 10.1145/3485447.3512140. URL: https://doi.org/10.1145/3485447.3512140.

Araci, Dogu (2019). "Finbert: Financial sentiment analysis with pre-trained language models". In: *arXiv preprint arXiv:1908.10063*.

El-Beltagy, Samhaa R. and Ahmed Rafea (2009). "KP-Miner: A keyphrase extraction system for English and Arabic documents". In: *Information Systems* 34 (1). ISSN: 03064379. DOI: 10.1016/j.is.2008.05.002.

Bennani-Smires, Kamil et al. (2018). "Simple unsupervised keyphrase extraction using sentence embeddings". In: DOI: 10.18653/v1/k18-1022.

Blei, David M, Andrew Y Ng, and Michael I Jordan (2003). "Latent dirichlet allocation". In: *Journal of machine Learning research* 3.Jan, pp. 993–1022.

Boleda, Gemma (2020). "Distributional semantics and linguistic theory". In: *Annual Review of Linguistics* 6, pp. 213–234.

Bougouin, Adrien, Florian Boudin, and Béatrice Daille (2013). "TopicRank: Graph-Based Topic Ranking for Keyphrase Extraction". In.

Campos, Ricardo et al. (2020). "YAKE! Keyword extraction from single documents using multiple local features". In: *Information Sciences* 509. ISSN: 00200255. DOI: 10.1016/j.ins.2019.09.013.

Caragea, Cornelia et al. (2014). "Citation-enhanced keyphrase extraction from research papers: A supervised approach". In: DOI: 10.3115/v1/d14-1150.

Chan, Hou Pong et al. (2020). "Neural keyphrase generation via reinforcement learning with adaptive rewards". In: DOI: 10.18653/v1/p19-1208.

Chronis, Gabriella and Katrin Erk (Nov. 2020). "When is a bishop not like a rook? When it's like a rabbi! Multi-prototype BERT embeddings for estimating semantic relationships". In: *Proceedings of the 24th Conference on Computational Natural Language Learning*. Online: Association for Computational Linguistics,

WWW.UPDF.COM

pp. 227–244. DOI: 10.18653/v1/2020.conll-1.17. URL: https://aclanthology.org/2020.conll-1.17.

Church, Kenneth Ward and Valia Kordoni (2022). “Emerging Trends: SOTA-Chasing”. In: *Natural Language Engineering* 28.2, pp. 249–269. DOI: 10.1017/S1351324922000043.

Dalgaard, Peter (2020). *Introductory statistics with R*. Springer.

Dekking, Frederik Michel et al. (2005). *A Modern Introduction to Probability and Statistics: Understanding why and how*. Vol. 488. Springer.

Devlin, Jacob et al. (2018). “Bert: Pre-training of deep bidirectional transformers for language understanding”. In: *arXiv preprint arXiv:1810.04805*.

Diez, David M, Christopher D Barr, and Mine Cetinkaya-Rundel (2012). *OpenIntro statistics*. OpenIntro Boston, MA, USA:

Eminagaoglu, Mete (2022). “A new similarity measure for vector space models in text classification and information retrieval”. In: *Journal of Information Science* 48.4, pp. 463–476.

Faber, Michael Havbro (2012). *Statistics and Probability Theory. In Pursuit of Engineering Decision Support*. Springer Dordrecht. DOI: https://doi-org.proxy.library.uu.nl/10.1007/978-94-007-4056-3.

Florescu, Corina and Cornelia Caragea (2017). “PositionRank: An unsupervised approach to keyphrase extraction from scholarly documents”. In: vol. 1. DOI: 10.18653/v1/P17-1102.

Garrido-Muñoz, Ismael et al. (2021). “A survey on bias in deep NLP”. In: *Applied Sciences* 11.7, p. 3184.

Gebru, Timnit et al. (2021). “Datasheets for datasets”. In: *Communications of the ACM* 64.12, pp. 86–92.

Gollapalli, Sujatha Das, Xiao-Li Li, and Peng Yang (2017). “Incorporating expert knowledge into keyphrase extraction”. In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1.

Gu, Yu et al. (2021). “Domain-specific language model pretraining for biomedical natural language processing”. In: *ACM Transactions on Computing for Healthcare (HEALTH)* 3.1, pp. 1–23.

Guerra, Julio, Sergey Sosnovsky, and Peter Brusilovsky (2013). “When one textbook is not enough: Linking multiple textbooks using probabilistic topic models”. In: *European conference on technology enhanced learning*. Springer, pp. 125–138.

Hasan, Kazi Saidul and Vincent Ng (2014). “Automatic keyphrase extraction: A survey of the state of the art”. In: vol. 1. DOI: 10.3115/v1/p14-1119.

Jay L. Devore, Kenneth N. Berk (2014). *Modern Mathematical Statistics with Applications*. Springer New York, NY. DOI: https://doi-org.proxy.library.uu.nl/10.1007/978-1-4614-0391-3.

Jiang, Xin, Yunhua Hu, and Hang Li (2009). “A ranking approach to keyphrase extraction”. In: DOI: 10.1145/1571941.1572113.

Kaltenbach, Hans-Michael (2007). *A Concise Guide to Statistics*. Springer Berlin, Heidelberg. DOI: https://doi-org.proxy.library.uu.nl/10.1007/978-3-642-23502-3.

Khan, Muhammad Qasim et al. (2022). “Impact analysis of keyword extraction using contextual word embedding”. In: *PeerJ Computer Science* 8, e967.

Kingma, Diederik P and Jimmy Ba (2014). “Adam: A method for stochastic optimization”. In: *arXiv preprint arXiv:1412.6980*.

WWW.UPDF.COM

Koch, Bernard et al. (2021). *Reduced, Reused and Recycled: The Life of a Dataset in Machine Learning Research*. DOI: 10.48550/ARXIV.2112.01716. URL: https://arxiv.org/abs/2112.01716.

Lee, Dik L, Huei Chuang, and Kent Seamons (1997). “Document ranking and the vector-space model”. In: *IEEE software* 14.2, pp. 67–75.

Lenci, Alessandro (2018). “Distributional models of word meaning”. In: *Annual review of Linguistics* 4, pp. 151–171.

Lenci, Alessandro et al. (2022). “A comparative evaluation and analysis of three generations of Distributional Semantic Models”. In: *Language Resources and Evaluation*, pp. 1–45.

Li, Piji (2020). “An empirical investigation of pre-trained transformer language models for open-domain dialogue generation”. In: *arXiv preprint arXiv:2003.04195*.

Liu, Zhiyuan, Wenyi Huang, et al. (2010). “Automatic keyphrase extraction via topic decomposition”. In.

Liu, Zhiyuan, Peng Li, et al. (2009). “Clustering to find exemplar terms for keyphrase extraction”. In: DOI: 10.3115/1699510.1699544.

Liu, Zhiyuan, Yankai Lin, and Maosong Sun (2020). “Document Representation”. In: *Representation Learning for Natural Language Processing*. Singapore: Springer Singapore, pp. 91–123. ISBN: 978-981-15-5573-2. DOI: 10.1007/978-981-15-5573-2_5. URL: https://doi.org/10.1007/978-981-15-5573-2_5.

Lv, Yuanhua and ChengXiang Zhai (2011). “When Documents Are Very Long, BM25 Fails!” In: *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*. SIGIR ’11. Beijing, China: Association for Computing Machinery, pp. 1103–1104. ISBN: 9781450307574. DOI: 10.1145/2009916.2010070. URL: https://doi.org/10.1145/2009916.2010070.

Madsen, Birger Stjernholm (2016). *Statistics for Non-Statisticians*. Springer Berlin, Heidelberg. DOI: https://doi-org.proxy.library.uu.nl/10.1007/978-3-662-49349-6.

Martinc, Matej, Blaž Škrlj, and Senja Pollak (2021). “TNT-KID: Transformer-based neural tagger for keyword identification”. In: *Natural Language Engineering*. ISSN: 14698110. DOI: 10.1017/S1351324921000127.

McKnight, Patrick E and Julius Najab (2010). “Mann-Whitney U Test”. In: *The Corsini encyclopedia of psychology*, pp. 1–1.

Meng, Rui et al. (2017). “Deep keyphrase generation”. In: vol. 1. DOI: 10.18653/v1/P17-1054.

Mickus, Timothee et al. (2019). “What do you mean, BERT? Assessing BERT as a Distributional Semantics Model”. In: *arXiv preprint arXiv:1911.05758*.

Mihalcea, Rada and Paul Tarau (2004). “Textrank: Bringing order into text”. In: *Proceedings of the 2004 conference on empirical methods in natural language processing*, pp. 404–411.

Mulvany, Nancy C. (2013). *Indexing Books, Second Edition*. DOI: 10.7208/chicago/9780226550176.001.0001.

Nayak, Pandu (Oct. 2019). *Understanding searches better than ever before*. URL: https://blog.google/products/search/search-language-understanding-bert/.

Page, Lawrence and Sergey Brin (1998). “The anatomy of a large-scale hypertextual Web search engine”. In: *Computer Networks* 30 (1-7). ISSN: 13891286. DOI: 10.1016/s0169-7552(98)00110-x.

Papagiannopoulou, Eirini and Grigorios Tsoumakas (2020). *A review of keyphrase extraction*. DOI: 10.1002/widm.1339.

Paperno, Denis et al. (2016). “The LAMBADA dataset: Word prediction requiring a broad discourse context”. In: *arXiv preprint arXiv:1606.06031*.

Poerner, Nina, Ulli Waltinger, and Hinrich Schütze (2019). “E-BERT: Efficient-yet-effective entity embeddings for BERT”. In: *arXiv preprint arXiv:1911.03681*.

Robertson, Stephen, Hugo Zaragoza, et al. (2009). “The probabilistic relevance framework: BM25 and beyond”. In: *Foundations and Trends® in Information Retrieval* 3.4, pp. 333–389.

Rogers, Anna, Olga Kovaleva, and Anna Rumshisky (2020). “A primer in bertology: What we know about how bert works”. In: *Transactions of the Association for Computational Linguistics* 8, pp. 842–866.

Sahrawat, Dhruva et al. (2020). “Keyphrase extraction as sequence labeling using contextualized embeddings”. In: vol. 12036 LNCS. DOI: 10.1007/978-3-030-45442-5_41.

Salton, G., A. Wong, and C. S. Yang (Nov. 1975). “A Vector Space Model for Automatic Indexing”. In: *Commun. ACM* 18.11, pp. 613–620. ISSN: 0001-0782. DOI: 10.1145/361219.361220. URL: https://doi.org/10.1145/361219.361220.

Salton, Gerard and Michael J McGill (1983). *Introduction to modern information retrieval*. mcgraw-hill.

Sanh, Victor et al. (2019). “DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter”. In: *arXiv preprint arXiv:1910.01108*.

Saracevic, Tefko (2007). “Relevance: A review of the literature and a framework for thinking on the notion in information science. Part II: Nature and manifestations of relevance”. In: *Journal of the American society for information science and technology* 58.13, pp. 1915–1933.

Shahmirzadi, Omid, Adam Lugowski, and Kenneth Younge (2019). “Text similarity in vector space models: a comparative study”. In: *2019 18th IEEE international conference on machine learning and applications (ICMLA)*. IEEE, pp. 659–666.

Shanmugam, Ramalingam and Rajan Chattamvelli (2015). *Statistics for scientists and engineers*. John Wiley & Sons, Incorporated.

Shi, Wei et al. (2017). “Keyphrase Extraction Using Knowledge Graphs”. In: *Data Science and Engineering* 2 (4). ISSN: 23641541. DOI: 10.1007/s41019-017-0055-z.

Srivastava, Nitish et al. (2014). “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1, pp. 1929–1958.

Sun, Chengyu et al. (2020). “A review of unsupervised keyphrase extraction methods using within-collection resources”. In: *Symmetry* 12 (11). ISSN: 20738994. DOI: 10.3390/sym12111864.

Tomokiyo, Takashi and Matthew Hurst (2003). “A language model approach to keyphrase extraction”. In: DOI: 10.3115/1119282.1119287.

Vaswani, Ashish et al. (2017). “Attention is all you need”. In: *Advances in neural information processing systems* 30.

UPDF
WWW.UPDF.COM

Vulić, Ivan et al. (Nov. 2020). “Probing Pretrained Language Models for Lexical Semantics”. In: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Online: Association for Computational Linguistics, pp. 7222–7240. DOI: 10.18653/v1/2020.emnlp-main.586. URL: https://aclanthology.org/2020.emnlp-main.586.

Walpole, Ronald E et al. (1993). *Probability and statistics for engineers and scientists*. Vol. 5. Macmillan New York.

Wan, Xiaojun and Jianguo Xiao (2008). “Single document keyphrase extraction using neighborhood knowledge”. In: vol. 2.

Wang, Alex et al. (2018). “GLUE: A multi-task benchmark and analysis platform for natural language understanding”. In: *arXiv preprint arXiv:1804.07461*.

Wang, Shuting et al. (2015). “Concept hierarchy extraction from textbooks”. In: DOI: 10.1145/2682571.2797062.

Wikimedia (2017a). *Long Short-Term Memory*. URL: https://upload.wikimedia.org/wikipedia/commons/5/5f/Gated_Recurrent_Unit.svg.

— (2017b). *$Recurrent_neural_network$*. URL: https://en.wikipedia.org/wiki/Recurrent_neural_network#/media/File:Recurrent_neural_network_unfold.svg.

Witten, Ian H. et al. (1999). *KEA: Practical Automatic Keyphrase Extraction*. arXiv: cs/9902007 [cs.DL].

Wu, Yonghui et al. (2016). “Google’s neural machine translation system: Bridging the gap between human and machine translation”. In: *arXiv preprint arXiv:1609.08144*.

Yu, Yang and Vincent Ng (2018). “Wikirank: Improving keyphrase extraction based on background knowledge”. In: *arXiv preprint arXiv:1803.09000*.

Zhang, Qi et al. (2016). “Keyphrase extraction using deep recurrent neural networks on twitter”. In: DOI: 10.18653/v1/d16-1080.

Zhang, Yin, Rong Jin, and Zhi-Hua Zhou (2010). “Understanding bag-of-words model: a statistical framework”. In: *International journal of machine learning and cybernetics* 1.1, pp. 43–52.

Zheng, Zhe et al. (2022). “Pretrained Domain-Specific Language Model for General Information Retrieval Tasks in the AEC Domain”. In: *arXiv preprint arXiv:2203.04729*.

Zhou, Sicheng et al. (2022). “CancerBERT: a cancer domain-specific language model for extracting breast cancer phenotypes from electronic health records”. In: *Journal of the American Medical Informatics Association*.

Zhu, Yukun et al. (2015). “Aligning books and movies: Towards story-like visual explanations by watching movies and reading books”. In: *Proceedings of the IEEE international conference on computer vision*, pp. 19–27.

UPDF
WWW.UPDF.COM

Appendix A

Indexer Interview

This section contains two interviews conducted with professional indexers to gain more about their job, what are the step behind the creation of indexes, and how humans approach this task.

Interviewer: How does your job start? What do you do when you receive a book that has to be indexed?

Interviewee: So what I get is the final PDF of the book. (. . .) I actually print things out, I do things from print, I've been doing this a long time. So I always print the press out, but then I will just start reading it. I know generally what the book is going to be about. When I start reading it, I use very fashion highlights so I will go through and highlight things that I think are important.

Interviewer: This is connected to a question that I wanted to ask: do you use external thanks sources of information or do you just rely on what you know about the topic? for example, if you're not sure if something is relevant do you check somewhere?

Interviewee: Yeah, the bulk of my work is medical and scientific. Sometimes if I don't recognize something or I think it might be spelled incorrectly or something I will do a check on Google, and often Wikipedia is the thing that comes up, but yes I will use that to check if it's not clear in the text what when something category belongs something. I actually read the whole thing first before I start indexing anything because what happens is that something that seems really important at the beginning, (. . .) might be a small topic. So the first chapters are very difficult because you tend to overindex that because you are very involved in the subject.

Interviewer: So it really depends on the focus of the single book. From what you said, it seems to me that some terms can be relevant in some books but not relevant in some others so it depends on the focus of their single chapter. Do you think there are some universal concepts that must be included in the index?

Interviewee: Yes, for example in medical books names of diseases. Drugs although not all the time because you could end up with just a whole list of paracetamol or something, those sorts of things yes you put in. What you want to focus on depends a bit on what the book is about. So if you go to a book about pain, what you don't want is to pick out everything every time it says pain. Otherwise, you just have a huge long list of "pain". (. . .) we called the meta topic which is the main thing

UPDF
WWW.UPDF.COM

the book is about. The general indexing is that you don't actually index that topic. Because the whole book is about it. So in a book about "pain", you would not have an index about pain because that would be the whole book basically. You know it also depends how often something's mentioned. So something can be mentioned once and it could cover a page of a really important, something can be mentioned once in a bit of a sentence, in the list and it's not important. That's when the context becomes relevant.

Interviewer: Also, another thing that I was not sure about is the role of headings at this point, the main headings more specifically because at this point as you're saying they usually don't cover the main topic. But they represent important concepts inside the book and these concepts are related to that. So what is the role of main headings? And what is the role of subheadings?

Interviewee: This is why you need knowledge of the subject. So for instance I don't do Law or Language books, History I don't do those, so I don't know what somebody will want to look up. So you need to have knowledge of the subject and know what somebody's going to look up. You can tell if you look at what was being indexed by somebody who doesn't know the subject (. . .) so you need to know the main things. They are also quite specific so you can't have a main heading "pain" that is too general. Subheadings are used if the main heading needs some explanation. Think about statistics. . . statistical tests. If it was a medical book you might have main headings if a statistical test as a subheading. f it was a book on statistics, those would have their own headings. So context is a lot to do with it. . . So I'm trying to think about a better example. For instance, if you go to the index term "disease", you might find under the disease multiple sclerosis, you'd put diagnosis, treatment all that sort of related. Also, it is not useful to put a lot of subheadings with the same page numbers. So if you have: multiple "sclerosis" 221-222. you don't want a load of subheadings with the same page because that's no use to anybody, because they are just directing you to the same page. So it needs to be some concrete information in the subheadings. some live subheadings.

Interviewer: (. . .) Yes, so what else. Mmhm. . . in general how long does it take to you to produce an index. So from the beginning to the end how long does it take to you to finish?

Interviewee: Well of course it depends on how long the book is.. you also have to remember that this is a commercial exercise, so we get paid per page sometimes some people pay more than other people, so that means if you pay less you'll get a less detailed index. I'll just do a calculation actually. . . with a book of 300 pages which I might get paid 400 pounds and I divide by (. . .) So for a 300-page book on an I would probably spend about 15 hours. Something like that, but it also depends on the focus of the book.

Interviewer: Really. . . I thought it was taking longer to be honest because. Probably because to know what you're working on you already have that knowledge to index the book. So you don't have to study the subject from scratch.

Interviewee: Yeah, I wouldn't go to index a book about Law because I don't know anything about it. So I guess 300 pages about 15 hours. Remember that you don't read it like you read a book. Alright, you read it very quickly because you can skim over the important things. I'm a lot slower reading a novel than I am with a textbook

Interviewer: One last thing is that I noticed that some main headings sometimes are not concepts of Statistics. For example, I found one saying "black cherry tree example" or in another one there was "Mario kart". It's a bit difficult because if you want to create a model that extracts only the statistics-related terms then all these other main headings are not useful. So I guess you include those headings if you think that that exercise is particularly useful for the reader.

Interviewee: I thinking about that… I was doing a medical today it was something like "seat belts crush". But you know I always think who's reading the book, and they want to find something like. Actually, if your program is looking for terms you can't restrict to subject terms because other things, specifically like examples, are going to come up. I think that's probably true in most subjects to find words that wouldn't seem to be to do with the subject are actually there.

Interviewer: yeah, those are the ones that I'm struggling with because these machines can recognize if it is a statistical term or not but they are not aware of the book or the chapter as a whole. I'm still not able to inform the machine about that: what is the topic of the whole chapter, but this example might be useful to explain what is not. Well, I think that was the last question, so thanks again, everything was really helpful.

Interviewee: If you need any element please get in touch and if you need anybody else please let me know. Well, good luck with everything.

UPDF WWW.UPDF.COM

Appendix B

Heatmap Outliers

B.1 Cosine Similarities between Domain Related Terms

Table B.1: The domain-related index terms that after fine-tuning had lower cosine similarity. The total number is 21. 11 of these had a negative value after training.

Terms	**Cos. Sim. Before**	**Cos. Sim. After**	**N° of sentences**
algorithm	0.663	0.010	39
experiment wise error rate	0.730	0.220	1
graphics	0.661	-0.0790	75
consistency	0.654	0.302	5
p value	0.724	-0.702	100
delta method	0.697	-0.295	1
cook's distance	0.635	-0.676	2
u chart	0.667	0.600	1
multi modal	0.703	0.295	2
curve	0.696	-0.054	100
quantitative	0.708	-0.434	24
error bars	0.721	0.003	5
stem and leaf display	0.634	-0.023	17
response surface methodology	0.652	-0.754	1
aggregate	0.730	0.179	2
convenience samples	0.685	0.534	2
characteristic value	0.761	0.485	3
repeated measures designs	0.683	-0.786	1
logarithmic distribution	0.751	-0.189	2
q value	0.711	-0.773	7
line chart	0.647	0.546	1

UPDF
WWW.UPDF.COM

B.2 Cosine Similarities between Domain Related and Out-Of-Domain Terms

Terms	**Cos. Sim. Before**	**Cos. Sim. After**	**N° of sentences**
challenger example	0.415	0.993	3
spatial data	0.698	0.072	3
noise	0.660	0.963	13
process	0.679	0.964	100

Table B.2: The out-of-domain index terms that after fine-tuning had higher cosine similarity. The total number is 4. 3 of these had values above 95%.

UPDF
WWW.UPDF.COM

Appendix C

Novel Terms Extraction

Table C.1: The fine-tuned BERT was input with 24 sentences that contained statistical terms not seen during the training phase (written in bold characters). The ✓and ✗indicate if the term was recognized or not, respectively. Additionally, all the keywords extracted by the model are reported. For the examined sentences the accuracy was 37.5%.

Sentence	**Keywords Extracted**	
The **ridge regression** is a method of estimating the coefficients of multiple-regression models in scenarios where the independent variables are highly correlated. It has been used in many fields including econometrics, chemistry, and engineering.	method, coefficients, correlated, regression, independent variables	✗
The **Rao Score test** is an alternative to Likelihood Ratio and Wald tests.	test, tests, likelihood ratio	✗
In this article, we introduce a **bivariate sign test** for the one-sample bivariate location model using a bivariate ranked set sample (BVRSS).	set sample, bivariate sign test, sample bivariate location model, bivariate	✓
The **maximum entropy principle** states that the probability distribution which best represents the current state of knowledge about a system is the one with largest entropy, in the context of precisely stated prior data (such as a proposition that expresses testable information).	test, data, probability distribution, entropy, information	✗
In statistics, the **Wishart distribution** is a generalization to multiple dimensions of the gamma distribution. It is named in honor of John Wishart, who first formulated the distribution in 1928.	distribution, wishart distribution, statistics, gamma distribution	✓
The **alpha error** is made in testing a hypothesis when it is concluded that a result is positive, but it really is not. Also known as false positive.	error, hypothesis, false positive	✗
This article deals with **residual treatment effect** designed for the purpose of comparing v test treatments with a control treatment when the number of periods is no larger than v + 1.	test, residual treatment, control	✗

UPDF
WWW.UPDF.COM

In probability theory, statistics and econometrics, the Burr Type XII distribution or simply the **Burr distribution** is a continuous probability distribution for a non-negative random variable.	statistics, random variable, probability theory, continuous probability distribution, distribution	✗
In mathematics, **random graph** is the general term to refer to probability distributions over graphs. Random graphs may be described simply by a probability distribution, or by a random process which generates them.	random process, random graphs, probability distributions, random graph, probability distribution	✓
In statistics, a **multivariate Pareto distribution** is a multivariate extension of a univariate Pareto distribution.	pareto distribution, statistics	✗
In statistics, **principal component regression** (PCR) is a regression analysis technique that is based on principal component analysis (PCA).	analysis, statistics, regression analysis, regression	✗
In probability theory and statistics, the **negative multinomial distribution** is a generalization of the negative binomial distribution (NB(x0, p)) to more than two outcomes.	outcomes, statistics, negative multinomial distribution, probability theory, negative binomial distribution	✓
In statistical theory, a **pseudo-likelihood** is an approximation to the joint probability distribution of a collection of random variables.	random variables, pseudo - likelihood, joint probability distribution, approximation	✓
In statistics, a **k-statistic** is a minimum-variance unbiased estimator of a cumulant.	cumulant, unbiased estimator, statistics	✗
Concurrent deviation method is a very simple method of measuring correlation. Under this method, we consider only the directions of deviations.	method, method of measuring correlation, deviations, deviation method	✗
The **conditional logistic regression** is an extension of logistic regression that allows one to take into account stratification and matching.	conditional logistic regression, logistic regression	✓
A **concentration curve plot** is the cumulative percentage of the health variable (y- axis) against the cumulative percentage of the population, ranked by living standards, beginning with the poorest, and ending with the richest (x-axis).	percentage, population, concentration curve plot, variable	✓
In statistical significance testing, a **equal-tail test** is alternative ways of computing the statistical significance of a parameter inferred from a data set, in terms of a test statistic.	test, test statistic, statistical significance, statistical significance testing, data set, parameter	✗

UPDF WWW.UPDF.COM

In probability, statistics and related fields, a **Poisson point process** is a type of random mathematical object that consists of points randomly located on a mathematical space.	random, poisson point process, statistics, probability	✓
In statistics, **overdispersion** is the presence of greater variability (statistical dispersion) in a data set than would be expected based on a given statistical model.	statistical dispersion, statistics, variability, expected', statistical model, data set	✗
In probability theory, an **isotropic measure** is any mathematical measure that is invariant under linear isometries. It is a standard simplification and assumption used in probability theory.	probability theory, standard, linear	✗
The **joint-regression model** for two-way data assumes a linear relation between a continuous response and column effects.	joint - regression model, data, continuous, linear	✓
In some real data examples, the local likelihood estimation proves to be effective in uncovering nonlinear dependencies.	nonlinear dependencies, data, likelihood estimation	✗
The **Lomax distribution**, conditionally also called the Pareto Type II distribution, is a heavy-tail probability distribution used in business, economics, actuarial science, queueing theory and Internet traffic modeling.	lomax distribution, pareto type ii distribution, probability distribution, conditionally	✗

WWW.UPDF.COM

Appendix D

UMAP Projection of BERT's Word Space

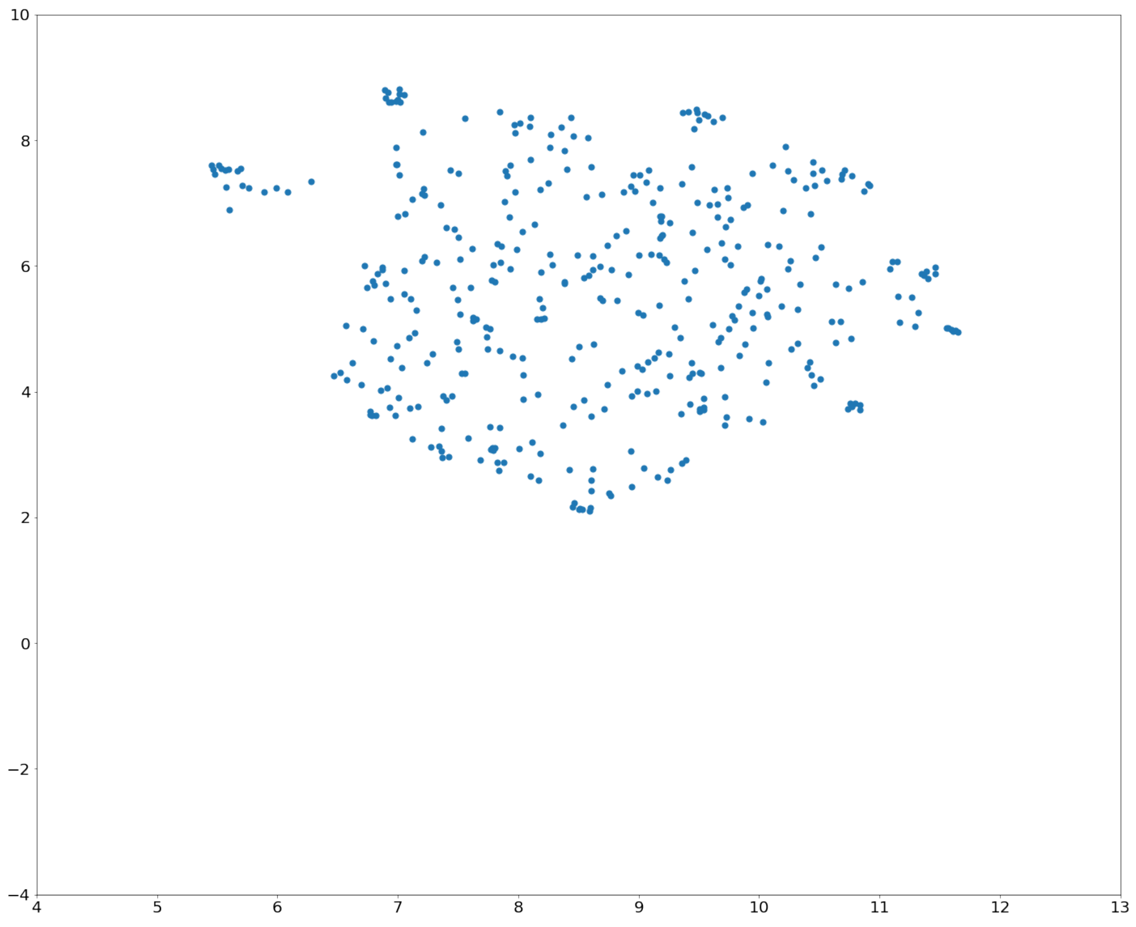


Figure D.1: Projection of domain-related terms from a pre-trained BERT.

WWW.UPDF.COM

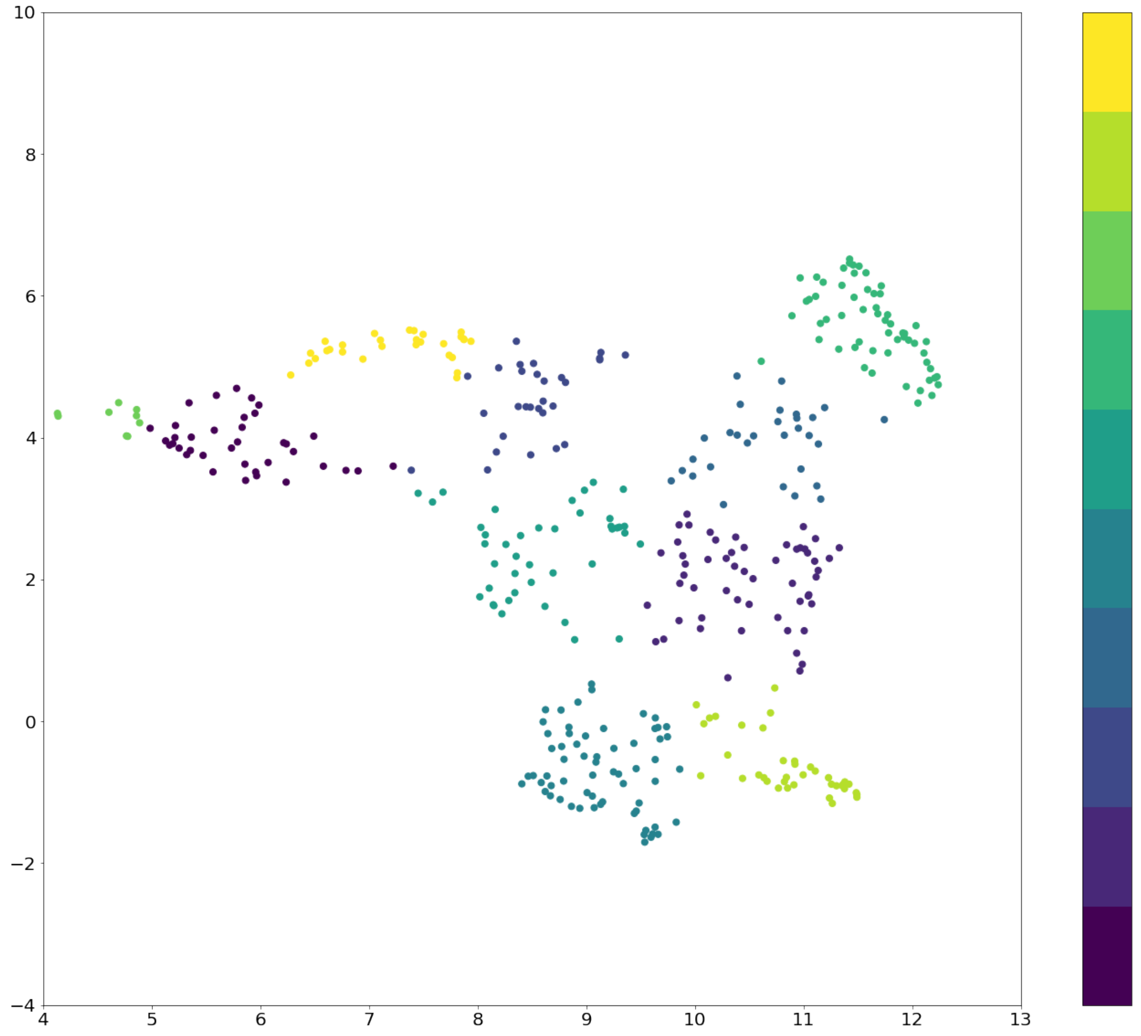


FIGURE D.2: Projection of domain-related terms after fine-tuning BERT on the target domain. Compared to the previous image, more defined clusters arise. Some of them are also highlighted using k-means clustering.

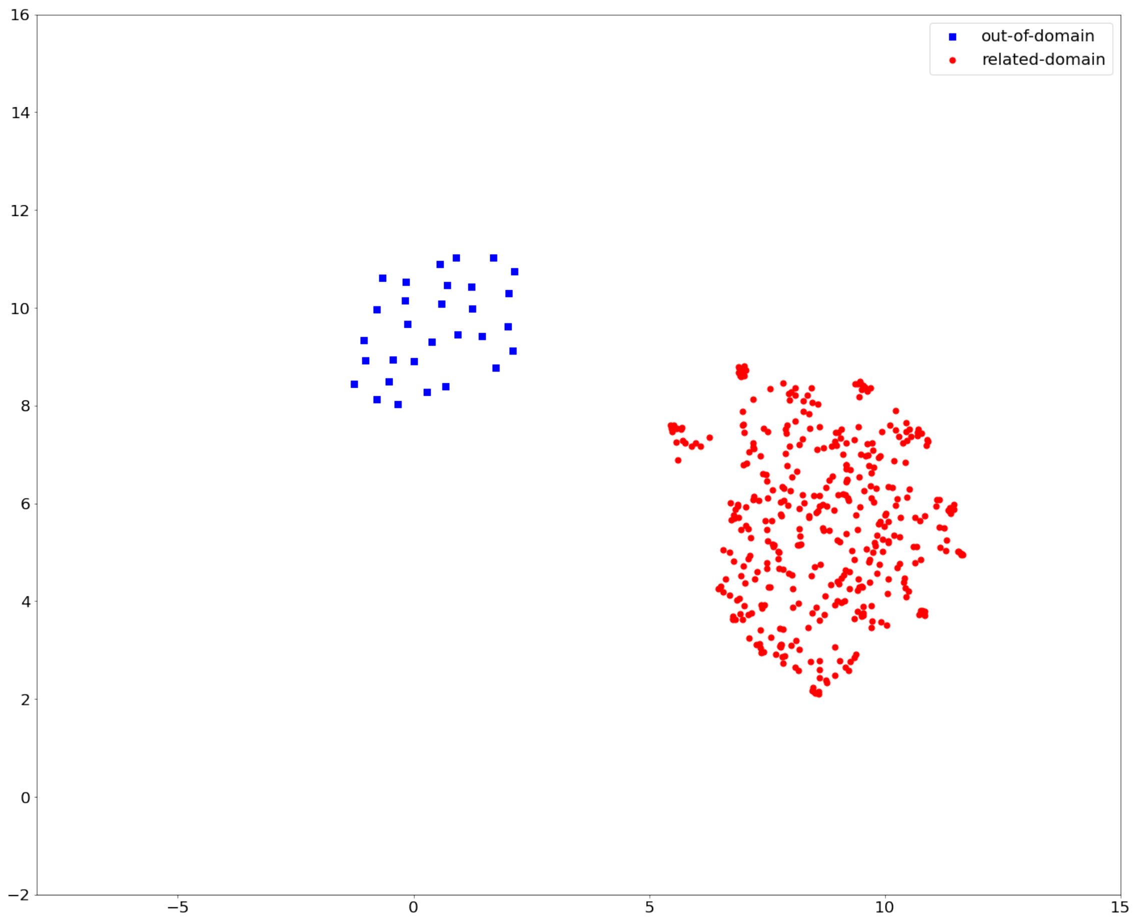


FIGURE D.3: Projection of domain-related and out-of-domain terms before fine-tuning BERT on the target domain.

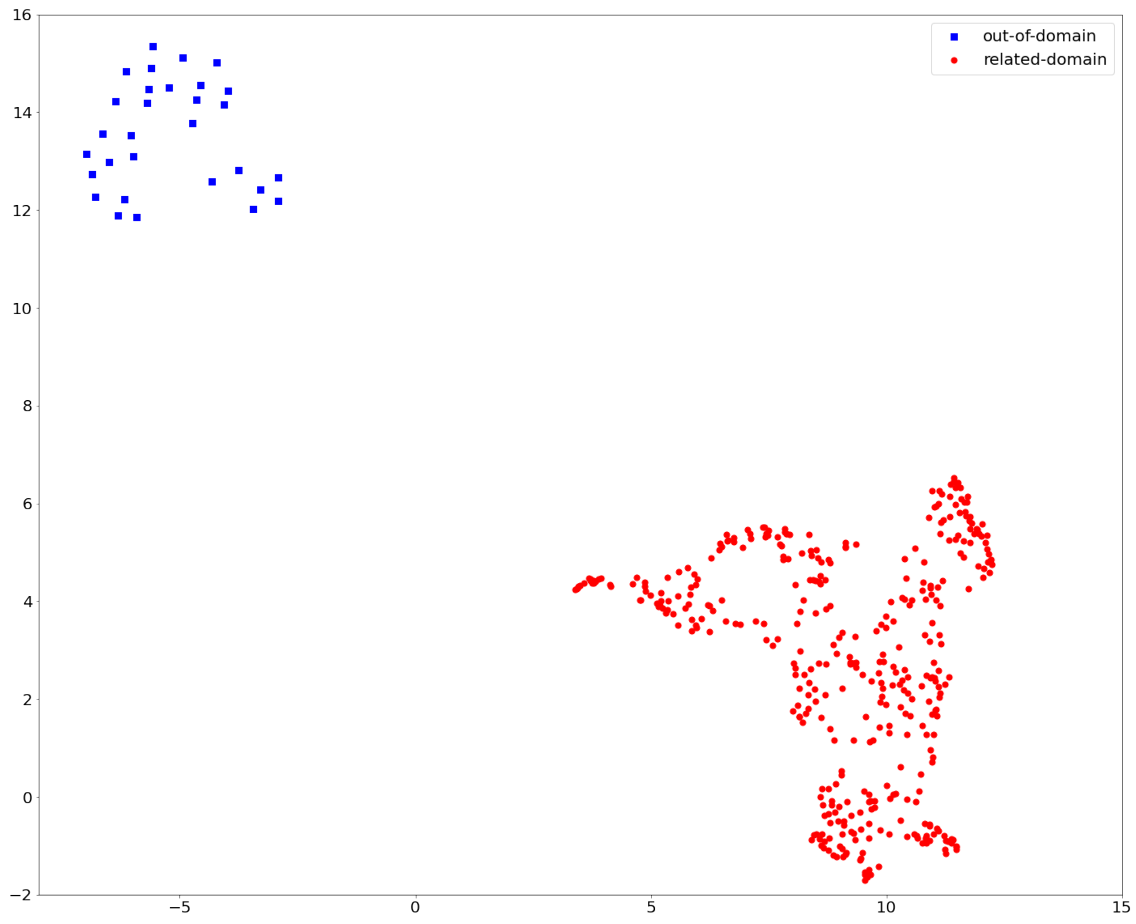


FIGURE D.4: Projection of domain-related and out-of-domain terms after fine-tuning BERT on the target domain: the space between the two clusters increased notably.